The Dissertation Committee for Subhashini Krishnasamy
certifies that this is the approved version of the following dissertation:

Online Learning and Decision-Making from Implicit Feedback

Committee:

Sanjay Shakkottai, Supervisor

Sriram Vishwanath, Co-Supervisor

François Baccelli

Gordan Žitković

Rayadurgam Srikant

Online Learning and Decision-Making from Implicit Feedback

by

Subhashini Krishnasamy, B.E., M.E.

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

May 2017

Acknowledgments

First, I wish to express a sincere thank you to Prof. Sanjay Shakkottai for the support, encouragement and advice he has provided throughout my time as a graduate student. I have been very fortunate to have an advisor who cares deeply about my work and who enthusiastically helps with any questions or problems I have. I am also grateful to Prof. Sriram Vishwanath for his continued support and encouragement. A special word of thanks goes to my mentors Urs Niesen and Piyush Gupta for their valuable guidance during my internship at Bell Labs.

I have greatly benefited from the courses, both technical and otherwise, that I took at UT; and for this, I am indebted to all the faculty members who taught them. I would like to thank the staff of WNCG and the ECE department, especially Karen Little and Melanie Gulick, for making administrative tasks at UT seem like a breeze.

Completing this thesis would not have been possible if not for the joint efforts of my collaborators – Siddhartha Banerjee, Sharayu Moharir, Rajat Sen, Prof. Sewoong Oh, Prof. Ramesh Johari, P. T. Akhil, Prof. Rajesh Sundaresan, Prof. Ari Arapostathis. I would specially like to thank Rajat for his perseverance and for the numerous times his skill for discovering useful tools helped in making headway in our research.

Thanks to all my friends and labmates at WNCG and UT, who made my

stay in Austin a memorable one. It is difficult to imagine life here without all your support and help. Finally, I must express my gratitude to my family for their patience and love all through.

Online Learning and Decision-Making from Implicit Feedback

Publication No. _____

Subhashini Krishnasamy, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Sanjay Shakkottai
Co-Supervisor: Sriram Vishwanath

This thesis focuses on designing learning and control algorithms for emerging resource allocation platforms like recommender systems, 5G wireless networks, and online marketplaces. These systems have an environment which is only partially known. Thus, the controllers need to make resource allocation decisions based on implicit feedback obtained from the environment based on past actions. The goal is to sequentially select actions using incremental feedback so as to optimize performance while simultaneously learning about the environment. We study three problems which exemplify this setting.

The first is an inference problem which requires identification of sponsored content in recommender systems. Specifically, we ask if it is possible to detect the existence of sponsored content disguised as genuine recommendations using implicit feedback from a subset of users of the recommender system.

The second problem is the design of scheduling algorithms for switch networks when the user-server link statistics are unknown (for e.g., in wireless networks,

online marketplaces). The scheduling algorithm has to tradeoff between scheduling the optimal links and obtaining sufficient feedback about all the links for accurate estimates. We observe the close connection of this problem to the stochastic multi-armed bandit problem and analyze bandit-style explore-exploit algorithms for learning the statistical parameters while simultaneously assigning servers to users.

The third is the joint problem of base station activation and rate allocation in an energy efficient wireless network when the channel statistics are unknown. The controller observes instantaneous channel rates of activated BSs, and thereby sequentially obtains implicit feedback about the channel. Here again, there is a tradeoff between learning the channel versus optimizing the operation cost based on estimated parameters.

For each of these systems, we propose algorithms with provable asymptotic guarantees. These learning algorithms highlight the use of implicit feedback in online decision making and control.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis studies online optimization problems at the intersection of machine learning and resource allocation. Online optimization problems find applications in various domains of computer science, operations research and economics. In many real world problems involving resource allocation, information about the environment is not completely known ahead of time but revealed only incrementally with time as the decisions are made. For example, a flight pricing algorithm has to gauge demand in an online fashion as customers buy tickets. A data center must allocate its resources dynamically based on arriving workload. Similar problems appear in other applications such as ad-space auctions, recommender systems, appointment scheduling etc. Usually, the goal is optimize a metric indicative of the system's long-term performance.

In many of these problems, the feedback data that arrives sequentially does not explicitly reveal the unknown factors about the environment but only gives *some* information about them. The decision maker has to extract useful information about the environment implicit in this data to make further decisions. Moreover, the decisions could influence not only the system's performance but also the feedback obtained subsequently. In such situations, the decision maker is faced with the dilemma of choosing actions that resolve uncertainty about the environment versus

actions that optimize the system performance based on existing knowledge of the environment.

We study three such problems which model practical settings in various domains such as recommender systems, crowdsourcing, online marketplaces, wireless networks etc.

1. The first is that of detecting advertisements disguised as genuine recommendations in a recommender system. Here, we propose to use the effectiveness of the recommendations made to users as feedback data.

2. The second is an online scheduling problem in a bipartite graph of servers and queues which have arriving customers. It is assumed that the success probabilities of the links between different queues and servers are non-homogeneous and unknown. For every scheduling decision made, the controller receives feedback about the *success* of that decision, and the goal is to optimize a metric which reflects the waiting time of customers.

3. The third is a joint problem of dynamic base-station activation and rate allocation in an energy-efficient cellular network. Here, it is assumed that the controller does not know the channel statistics but obtains instantaneous channel rates of activated base-stations.

In each of these problems, to represent the uncertain environment in the system, we use discrete time stochastic models whose parameters are initially not known to the decision-maker. As a model for implicit feedback, we assume that

the controller sequentially receives data samples from the environment (according to the stochastic model) based on its past decision history. For example, in the recommender system problem, we use probabilistic models to differentiate the behavior of a system that makes genuine recommendation from that which shows ads. The implicit feedback used here is the knowledge of the effectiveness of the recommendations made dynamically to the users. Likewise, in the bipartite scheduling problem, we assume that every assignment of a queue-server link is successful with some fixed probability independent of all previous assignments. The scheduler is unaware of these success probabilities but receives feedback about the success of its assignments in every time-slot. The scheduler progressively gains more of these samples from which it can infer the success probabilities of the links. But note that the goal is not to just *learn* these success probabilities. Instead, it is to minimize the wait time of the customers in the system. Likewise, in the resource allocation problem for energy-efficient networks, the channel is not known to the controller and the instantaneous channel information constitutes the implicit feedback that the controller obtains in every time-slot.

For all the three problems, we propose online algorithms that play the explore-exploit tradeoff of dynamically estimating the unknown parameters in the model and optimizing the system performance. We show asymptotic guarantees for these algorithms using tools in stochastic analysis such as queueing theory, concentration bounds etc. Below, we give a brief description of the problems and the contributions made by this thesis.

1.1 Problem Settings and Contributions

1.1.1 Detecting Sponsored Recommendations

Personalized recommender systems have been of tremendous benefit both to consumers by helping them sift through a vast number of products, web-pages, and news items and to online services by helping them present their products better. Such systems however provide great opportunities for targeted advertisements through the display of ads alongside genuine recommendations. We consider a *biased* recommendation system where such ads are displayed without any tags (disguised as genuine recommendations), rendering them indistinguishable to the end user. We specifically focus on those *biased* systems that systematically favor a few items over other better or at least equally good items, contrary to what an *objective* or unbiased system would do. Our goal is to answer the following question:

Is it possible to identify a biased *system using implicit feedback from a small subset of users?*

Contributions of this Thesis:

We identify key properties that distinguish a *biased* recommender system from an *objective* (unbiased) one. For one, an *objective* recommender system gives higher preference to higher ranked items in its recommendations, while a *biased* system frequently recommends ads even if they are not well-rated. We formulate a probabilistic model for the two classes of recommender systems based on some universal principles satisfied by *objective* and *biased* recommenders.

We propose an algorithm that can detect distinguish between *objective* and

biased recommenders through statistical analysis on the users' feedback. The algorithm requires only binary information indicating whether a user was satisfied with each of the recommended item or not. This makes the algorithm practically applicable to real world issues such as identification of search engine bias and pharmaceutical lobbying. We study the algorithm's performance using statistical analysis. Such an analysis helps us in identifying the key factors that affect the algorithm's performance. Finally, we show simulation results on the algorithm's performance for various combinations of data sets and recommendation algorithms.

The model for recommender systems, the anomaly detection algorithm and the various factors that affect the algorithm's performance are discussed in detail in Chapter 2.

1.1.2 Bandit Algorithms for Queueing Systems

The second problem is that of dynamic scheduling of servers to queues in a $U \times K$ switch network with non-homogeneous service rates among the links. More specifically, we consider a discrete-time stochastic switch network with U users and K servers ($U \leq K$) with a single link between every user-server pair. The service rate for each link is i.i.d. Bernoulli across time-slots and independent of other links. The mean service rate could be different for different links. We study the design of scheduling algorithms when the link rates are unknown to the scheduler. This model can be applied to queueing and scheduling problems in various systems like wireless networks, crowdsourcing platforms, cloud-computing centers etc.

We introduce this scheduling problem as the *queueing bandit problem*, which

is a multi-dimensional analog of the standard stochastic multi-armed bandit problem with U users and K arms and with the regret defined as the expected *queue-length* difference between the scheduling policy and a genie policy that knows the link rates. We study how the explore-exploit trade-off in the traditional stochastic multi-armed bandit problem generalizes to the multi-dimensional queueing network problem. Specifically, we ask the following questions:

1. *What are the best possible regret bounds for the queueing network?*
2. *What are the key differentiating factors in the design of algorithms for the queueing bandit problem as compared to the standard MAB problem?*

Contributions of this Thesis:

We introduce *queue-regret* – a notion of regret that is appropriate for queueing systems. We then analyze a special case of the queueing bandit problem, which we call the case of *unique optimal matching*. A switching system is said to have a unique optimal matching if each user has a unique server which is stochastically the best for that user and these optimal user-server pairs form a perfect matching in the bipartite network.

For this special case, we derive lower bounds for the queue-regret for a well-known class of traditional bandit algorithms – the class of all α-consistent policies. We propose a scheduling algorithm for the case of unique optimal matching. The proposed algorithm is a variant of traditional bandit algorithm with the addition of more aggressive and structured exploration. We prove that the proposed algorithm

achieves the same scaling as the lower bound for the α-consistent class upto some poly-logarithmic factor.

Our analysis shows that considering queue-lengths as the metric for regret makes a fundamental difference in achievable bounds. In addition, we gain some useful insight into the right kind of exploration strategy for the queueing bandit problem. The model and the proposed algorithm are discussed in detail in Chapter 3.

1.1.3 Scheduling with Energy Costs

For the third part, we again consider a resource allocation problem in the domain of designing energy-efficient cellular networks. In small-cell wireless networks where users are connected to multiple base stations (BSs), it is often advantageous to opportunistically switch off a subset of BSs to minimize energy costs. Modern cellular standards provide a great opportunity to achieve low energy operation by enabling BSs to operate in *sleep* mode where the BSs can be switched OFF dynamically on a per-time-slot basis. Considering that the cost of operation includes both energy to maintain active BSs and to switch BSs from one state to another, we study the joint problem of BS activation and channel scheduling from a common bandwidth pool. The objective is to minimize the operation cost subject to maintaining stable user queues.

We address the following question:

What is a good activation-cum-scheduling algorithm when the scheduler has no knowledge of channel statistics but can only obtain instantaneous channel rates for active BSs in every time-slot?

Evidently, here too, we have to deal with the explore-exploit dilemma, where there is a conflict between activating diverse subsets of BSs to learn their channel statistics versus using estimated parameters to activate BSs so as to optimize operation cost and ensure queue stability.

Contributions of this Thesis:

We first consider the activation-cum-scheduling problem where the channel statistics are known to the scheduler. We observe that, even for this simpler problem, one cannot use greedy primal dual algorithms (with virtual queues), which are shown to be optimal in traditional stochastic network resource allocation problems using the standard Lyapunov technique. This is due to the existence of switching cost which introduces a dependence between activation states in consecutive time-slots. This is an important observation since it introduces a new class of problems for which existing methods are not applicable.

For this problem (where the channel statistics are known), we propose an algorithm that switches BS states only at a much slower time-scale than the channel scheduling. This ensures not only a low switching cost, but by appropriately choosing the active BSs at a slow time-scale and scheduling channels at a fast time-scale, we show that the proposed algorithm can achieve close to optimal total operation cost while maintaining queue stability.

For the setting where the channel statistics are unknown, we extend the above algorithm to include an ϵ-greedy type of explore-exploit strategy to learn the channel statistics. Again, we show that this extended algorithm can achieve close

to optimal cost while maintaining queue stability. A key challenge in proving queue stability is resolved using some new convergence results for time-inhomogeneous Markov chains. We believe that these technical results are of interest in their own right as they further extend existing results on this topic.

The system model and the proposed algorithm are described in detail in Chapter 4.

Chapter 2

Detecting Sponsored Recommendations

2.1 Introduction

The growth of online services has provided a vast variety of choices to users. This choice exists today in multiple domains including e-commerce with a variety of products, and online entertainment (NetFlix, Pandora). With users having to choose from an overwhelming set of items, recommender systems have become indispensable in easing the information overload and search complexity. Recommender systems are not restricted to retail businesses. A search engine like Google can be viewed as a recommendation engine that helps users find relevant information by ranking the search results according to their search criteria, history and other personal information. Social networking sites like Twitter and Facebook display Tweets and News Feed based on users' past behavior and their connections to other users. News portals like Yahoo! News also present personalized content to online news readers.

Personalized recommender systems serve as an attractive platform for advertisers to reach their targeted consumers[1]. It is now customary to see ads alongside

[1]Subhashini Krishnasamy, Rajat Sen, Sanjay Shakkottai, and Sewoong Oh. "Detecting sponsored recommendations". In ACM Trans. Model. Perform. Eval. Comput. Syst., 2(1):6:1–6:29, November 2016. Co-authors of the paper made equal contributions in obtaining these results.

other genuine recommendations in many of the websites that provide recommendation services. One can distinguish these ads from genuine recommendations, for example, by the location of their placement or by their special tags. But recommendation engines are not legally obliged to facilitate such distinction and could possibly serve these ads mixed with genuine recommendations in a manner that renders them indistinguishable to users. Such advertising is commonly known in the online marketing world as *native advertising*.

According to the Interactive Advertising Bureau (IAB) [5], native advertising aims *"to deliver paid ads that are so cohesive with the page content, assimilated into the design, and consistent with the platform behavior that the viewer simply feels that they belong."* Although native advertising is a relatively new phenomenon, it is expanding rapidly according to the numerous surveys conducted on sponsored content. According to a 2013 survey by native advertising exchange Hexagram and digital consultancy firm Spada [7], 62% of publishers offered native advertising at the time of survey and another 16% planned to do so within a year. A survey by eMarketer [6] reveals that marketing agencies see huge potential in native ads – almost three quarters of marketers already use native advertising and only 7% do not plan to use them in future. Spending on native advertising in the US is predicted to increase from 1.3 billion dollars in 2013 to 9.4 billion dollars in 2018. This explosion is fueling the growth of native advertising as a business ecosystem with marketing agencies as consumers, publishers as media providers and several companies such as Hexagram, HubSpot and Nudge offering tools to facilitate this exchange of digital media space. Indeed, several watchdog groups (both federal and

industry sponsored) are cautious of this trend and have issued guidelines (e.g., FTC policy [1], Interactive Advertising Bureau [5, 3]).

A *biased recommender system* which shows paid ads and sponsored content without being transparent about them can have far reaching consequences. One of these is user dissatisfaction with the recommendations [24]. A recent survey by Facebook shows that users find sponsored ads mixed with genuine posts in their News Feed more annoying than the explicit, well-separated ads [14]. Social and political consequences of bias in the context of media and online content have also been studied [49, 71, 50].

Thus, a basic question of interest to consumers is whether or not biased recommendations can be detected. It would be beneficial for watchdog agencies and groups that enforce compliance of ad disclosure policies [2, 4, 1] to have suitable tools to detect sponsored recommendations that do not have explicit tags. Furthermore, it could also be used by businesses to monitor their own products or their competitor's products. As an example, India's anti-trust watch-dog, Competition Commission of India (CCI), recently filed charges against Google for rigging its search results in favor of its own products and services [64]. The charges were based on investigation of complaints from various competitors spanning several businesses including search, social networks, e-commerce etc. Automated tools that indicate potential biases in recommendations would be of utility to these type of agencies.

Modern recommender systems, in general, consist of two components: (i) learns individual preferences from user feedback, and (ii) recommends items to users based on the estimated preferences. This combination of learning and recommending is

bound to be noisy (the learning phase will *explore* individual preferences typically by presenting "random" recommendations), and several recommendations to users will likely be ineffective. Critically, both noise and bias manifest as bad recommendations to users. However, noise is benign and is a consequence of learning, while bias is systematic and is to be deprecated.

The problem of detecting bias in its most general sense is a broad topic and out of the scope of this thesis. We focus on a detecting a specific type of bias as described above, where recommendation engines show sponsored content to users contrary to their preferences without being transparent about their recommendations. These biased systems *systematically favor a few items over other better or at least equally good items, contrary to what an objective or unbiased system would do.*

It should be noted that, with most service providers being non-transparent about their recommendation strategies, one cannot hope to know the exact statistical profile of the recommendation engine *a priori*. Therefore, the key is to identify the primary features that can be used to differentiate between the two types without any *a priori* knowledge about the particulars of the recommendation strategy. One could, for instance, consider the average rating or the average number of ineffective recommendations as the deciding criterion. However, as we also demonstrate through simulations, such a basic algorithm based solely on average performance cannot distinguish between deliberate systematic bias and innocuous random errors. This brings us to the key question: *Can we develop a better method to expose a biased recommender system?*

2.1.1 Contributions of this Work

We say a recommendation engine is *biased*, if it systematically favors a small set of items over other items in the database irrespective of users' preferences. On the other hand, we say that a recommendation engine is *objective*, if it satisfies a simple monotonic property in its recommendations to users – better suited items are given higher priority (in a statistical sense). The primary goal is to answer the following question: *Can a meaningful distinction be drawn between objective and biased recommendation engines?*

BiAD Algorithm: We propose an anomaly detection algorithm that we call *Bi*nary feedback *A*nomaly *D*etector (*BiAD*), which uses a statistical approach to identify a *biased* recommendation engine. Under appropriate conditions on the size of the ad-pool, the aggressiveness of the biased recommender system, and the number of users/samples, we show that BiAD correctly (with high probability) distinguishes between objective and biased recommendation engines.

The algorithm leverages user collaboration, and is based on the observation that a *biased* system is typically characterized by the occurrence of a *large number of ineffective recommendations in a small set of items*. On the contrary, giving higher priority to more effective items, as in an *objective* recommender system, precludes such concentration in a small set. Notably, since the users are not aware of the set of items, the BiAD algorithm is adaptive – as the recommender system learns users, the users "learn" the recommender system. Further, our algorithm relies only on binary feedback on the effectiveness of the recommendation. Although we assume in our model that this feedback is explicitly provided by the users, obtaining only implicit

feedback may be feasible in some practical scenarios. In such settings, a binary feedback model is extremely useful since it is easy to interpret implicit feedback in terms of effectiveness of the recommendation. In Section 2.4.3, we give example of a situation where only implicit feedback can be obtained and a binary model of effectiveness is applicable. Finally, the BiAD algorithm also works for a large class of recommender systems since our model does not place any constraints on the recommendation engine other than mild statistical conditions. We finally present extensive simulation results that cover various types of recommender systems and data sets to illustrate the wide applicability of the algorithm.

2.1.2 Related Work

Following the recent successes of the targeted advertising services, there have been several empirical studies that investigate the effects of displaying sponsored content alongside organic content [24, 57, 134]. It is empirically shown in [24] that customers are less likely to select recommendations which are tagged as "advertisement" or "sponsored", motivating the advertisers to remove such tags. There have also been attempts to explain such effects through theoretical models [143, 26]. In addition, several researchers have worked on designing systems and algorithms from the content provider's perspective for revenue maximization through efficient auction of the ad-space [90] and from the advertiser's perspective for effectively reaching the target audience [118, 135].

Our work, on the other hand, deals with identification of covert interaction between content providers and advertisers, specifically of promotional content being

passed off as editorial content or recommendations. This problem falls under the broad class of problems called *anomaly detection*, which generally involve identification of unusual patterns in a system. Examples of such problems include network intrusion detection, fraud detection, etc. [94, 30]. Prior work on anomaly detection in recommender systems exists from the perspective of a recommendation engine as a victim of false user-profile injections [34, 106]. To the best of our knowledge, ours is the first work that considers the problem from the users' perspective and proposes a mechanism for detection of bias in recommendation engines.

There is a vast literature in the broader context of anomaly detection – a comprehensive survey can be found in [38]. Various techniques of anomaly detection which include classification, clustering, statistical, information-theoretic methods etc. are applicable depending on the type of input data, the type of anomaly and the desired output. Clustering and neighborhood-based techniques have been proposed for many of the applications domains which seek to label a large collection of data points as either normal or anomalous [116, 113, 146, 68, 89]. These techniques are based on differentiation of features through comparison of multiple data points against each other. Another category of techniques called classification [121, 67, 22, 126, 102] is based on using training data – data points that have been labeled a priori – to learn the appropriate features of anomalous behavior.

The above mentioned techniques, designed for multiple input data instances, cannot be applied to problems like the current one, which requires detection of a single anomalous instance. For these problems, one has to rely on features specific to the problem to differentiate between normal and anomalous data instances. Our ap-

proach to the problem of detecting sponsored recommendations falls in the category of *statistical techniques* [12, 54, 11, 86, 144], following the classification of techniques in [38]. In this class of anomaly detection techniques, the features specific to the problem are characterized by a probabilistic model and statistical inference tests are used to determine whether or not the input instance conforms to the representative model.

In this setting, we make only mild assumptions in our model of the recommendation engine, and do not restrict to a specific type of recommendation algorithm. This enables our algorithm to be applied to a large class of recommender systems.

2.2 System Model

In this section, we describe our assumptions about the structural properties of *objective* and *biased* recommender systems by the means of a probabilistic model. This model does not include any particulars about the working of the recommendation engine and therefore typifies a broad class of recommender systems. Before we proceed to describe the model in detail, the salient features of this model are listed below:

- An *objective* recommendation engine has a fairly good estimate of the user preferences.
- An *objective* recommendation engine follows the monotonic property – higher preference to higher ranked items.

- A *biased* recommendation engine systematically gives preference to a small set of items irrespective of users' tastes.

Notation: Our notation $O, \Omega, \Theta, o, \omega$ to describe the asymptotics of various parameters with increasing size of the database (total number of items in the database) is according to the standard Landau notation. We say that an event occurs with high probability if the probability of the event tends to 1 as the size of the database goes to infinity. We use $\mathbb{1}\{\cdot\}$ to represent the indicator function, i.e.,

$$\mathbb{1}\{\mathsf{E}\} := \begin{cases} 1 & \text{if event } \mathsf{E} \text{ occurs,} \\ 0 & \text{otherwise.} \end{cases}$$

Equality and inequality between random variables always refer to almost sure (with probability 1) conditions unless otherwise specified. For example, if X and Y are two random variables, then $X = Y$ implies $X = Y$ *a.s.* For any given matrix, $\mathbf{R}$, the u^{th} row of $\mathbf{R}$ is represented by $\mathbf{R}_u$.

2.2.1 User-Item Database

The recommendation engine recommends products to users from a large database of m items indexed from 1 to m. A user's opinion about an item is represented by a numerical value that we call the user's *rating* of that item. It should be noted that these ratings are only an implicit representation of true opinions of the users – higher the rating, better suited is the item for the user. We denote the user-item rating matrix for the entire database by $\mathbf{R}$, where rows indicate users and columns indicate items. We introduce a parameter called the *efficacy threshold*, denoted by η which is used to represent opinions on a binary scale. We assume that

a user is satisfied with a recommendation if the rating of the recommended item is greater than or equal to η. We refer to such a recommendation and item as being *effective* for that user.

Definition 2.1 (Effective & Ineffective)**.** *An item i is effective for a user u if the rating of that item by the user, R_{ui} is at least η. Similarly, a recommendation is said to be effective for a user if the recommended item is effective. An item or recommendation that is not effective is said to be ineffective.*

Let $f_u(\eta, [m])$ denote the number of items in the database $[m]$ whose rating is greater than or equal to η for user u. In other words, it is the number of effective items in the database for user u.

Let us define the function $F : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}$ as follows:

$$F(r, \mathbf{R}_u) := |\{i : R_{ui} \geq r\}|,$$

where R_{ui} is the i^{th} element of the m-length vector $\mathbf{R}_u$. This function is used to find the number of items whose rating exceeds value r for any player u if the ratings of all the items in the database for player u is given by $\mathbf{R}_u$. For example, if $\mathbf{R}_u$ is the row corresponding to player u in the rating matrix, $\mathbf{R}$, then $F(\eta, \mathbf{R}_u)$ is equal to $f_u(\eta, [m])$, the number of effective items for user u. Similarly, $F(R_{ui}, \mathbf{R}_u)$ gives the rank of item i for user u. Also note that for any given $\mathbf{R}_u$, $F(r, \mathbf{R}_u)$ is a non-increasing function of r.

2.2.2 Recommendation Engine

We next describe the behavior of a recommendation engine using a probabilistic model. Let $\mathbb{1}_{ui}(t)$ indicate whether item i has been recommended to user u at time t, i.e.,

$$\mathbb{1}_{ui}(t) := \begin{cases} 1 & \text{if item } i \text{ is recommended to user } u \text{ at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

We make the following assumption about any recommender system: An item that has been recommended to a user once is not recommended to the same user again, i.e., for any user, u and item, i, $\sum_{t=1}^{\infty} \mathbb{1}_{ui}(t) \leq 1$.

2.2.2.1 Objective Recommendation Engine

An *objective* recommendation engine is considered to consist of two components - one is the *learning* strategy which estimates the user-item rating matrix by the means of available feedback from users, and another is the *recommendation* strategy which generates recommendations based on the estimated user preferences. Our model does not specify the details of the learning strategy except requiring that the output of the strategy, that is the estimate of the user-item matrix, be close to the original rating matrix, $\mathbf{R}$. Therefore, this model could be applied to a wide class of recommendation engines which estimate users' preferences fairly well. Let the estimate of the rating matrix at time t be denoted by $\hat{\mathbf{R}}(t) = \left[\hat{R}_{ui}(t)\right]$. This estimate is modeled as the sum of the original rating matrix and an additive noise matrix whose elements are independent across users, items and time. This can be written as $\hat{\mathbf{R}}(t) = \mathbf{R} + \boldsymbol{\epsilon}(t)$, where $\boldsymbol{\epsilon}(t) = [\epsilon_{ui}(t)]$ is the noise matrix and $\epsilon_{ui}(t)$ is independent of all other random variables for all u, u', i, i', t, t'.

The recommendation strategy uses the estimated user-item rating matrix $\hat{\mathbf{R}}(t)$ to make recommendations at time t. The following model characterizes the behavior of an *objective* recommendation strategy:

1. Recommendations are made based on a user-item weight matrix, denoted by $\mathbf{W}(t) = [W_{ui}(t)]$. This is a stochastic matrix (rows sum to one), which is updated based on the current estimate of the rating matrix, $\hat{\mathbf{R}}(t)$.

2. Given the weight matrix, a user is given a recommendation by choosing an item randomly, independent of everything else, with weights given by the row corresponding to the user in the user-item weight matrix.

3. At any time t, the weight matrix $\mathbf{W}(t)$ satisfies the following *monotonic* property: if i and j are two items that have not been shown to user u and the ratings are such that $\hat{R}_{ui}(t) \geq \hat{R}_{uj}(t)$, then the weights satisfy $W_{ui}(t) \geq W_{uj}(t)$.

2.2.2.2 Biased Recommendation Engine

A *biased* recommendation engine marks a small set of items, $\mathcal{A}$ ($\subseteq [m]$) from the item database as *ads*. To make a recommendation to a user, with probability γ, independent of everything else, it chooses an item that has not been shown from the ad-pool, $\mathcal{A}$. And with probability $1-\gamma$, it can follow any recommendation algorithm (for example, an *objective* recommendation algorithm). We refer to γ as the *bias probability*. Note that the strategy for showing ad items is unspecified except that no item is shown to a user twice. In particular, the engine may even customize its ad recommendations according to users' tastes. As in the case of the complete

database, let $f_u(\eta, \mathcal{A})$ denote the number of effective ads in the ad-pool, $\mathcal{A}$ for user, u.

2.2.3 Discussion of Assumptions

Some of the assumptions in the recommender system model above are present only for ease of analysis. We discuss below how they can be relaxed in practical settings.

1. It is assumed that, in any recommendation engine, an item once recommended to a user is not recommended to the same user again. This condition is required only to ensure that there are no repeated recommendations of sponsored advertisements that might be effective. Indeed, if all sponsored ad recommendations are effective, it would not be possible to distinguish them from genuine recommendations. This assumption can therefore be relaxed to require sufficient number of ineffective ad recommendations in a *biased* recommender system. (A precise mathematical statement is given at the end of this discussion.)

2. The noise in estimation of the user-item rating matrix is assumed to be additive i.i.d. noise. This can be replaced by a more general noise model in which the elements of the estimated user-item matrix are independent across users, items and time. The independence assumption is used to model arbitrary errors which are unlikely to skew the estimated matrix in such a way as to give high preference to a small number of ineffective items uniformly across a large subset of users.

3. We assume that a *biased* recommendation engine decides to show sponsored ads with probability γ (bias probability) independent of everything else. This assumption, again, is used only for ease of exposition. It is sufficient to have $\Omega(\gamma)$ fraction of the total recommendations from the ad-pool, not necessarily chosen at random.

To be mathematically precise, the assumptions discussed in points (1) and (3) can be relaxed to the following condition for the analytical results in this chapter to hold.

There exist constants $c_1, c_2 > 0$ such that for any $t_1 \geq 0, t_2 > t_1, n > 0$, and any set of users $\mathcal{U}$ such that $|\mathcal{U}| = n$,

$$\mathbb{P}\left[\sum_{l=t_1+1}^{t_2} \sum_{u\in\mathcal{U}, i\in\mathcal{A}} \mathbb{1}_{ui}(l) \geq c_2\gamma n(t_2 - t_1)\right] \geq 1 - e^{-c_1\gamma n}.$$

In other words, for any set of n users, if a biased *recommendation engine makes t recommendations to each user, then with probability at least $1 - e^{-c_1\gamma n}$, the total number of recommendations from the ad-pool among the nt recommendations is at least $c_2\gamma nt$.*

2.3 Anomaly Detection Algorithm and Theoretical Results

In this section, we describe the algorithm for detecting anomalous systems and provide analysis of Type I and Type II errors as in binary hypothesis testing.

2.3.1 Anomaly Detection System

The problem is to design a test to detect if a recommendation engine is *biased*. In other words, the test has to decide between the following two hypotheses:

- H_1 : "The recommendation engine is *biased*," and
- H_0 : "The recommendation engine is not *biased*."

It is similar to a hypothesis testing problem except that the statistical distribution for the two hypotheses are not well defined. The only *a priori* knowledge that is assumed is the structure of a *biased* recommendation engine as specified in Section 2.2.2. But the specifics of various parameters in the recommendation engine, such as bias probability γ and the ad-pool $\mathcal{A}$ is unknown. As in traditional hypothesis testing problems, we make use of multiple data points obtained from many users who constitute the *anomaly detection system*.

2.3.2 Feedback Data

The anomaly detection system consists of a set of n *players* which is a subset of the user database in the recommendation system. Without loss of generality, we denote these players as users indexed from 1 to n in the user database. These players give binary feedback (effective or ineffective) on the items recommended to them. The feedback can be subject to errors. We assume that the feedback given by user u about any recommendation is in error with probability ξ_u independent of everything else.

2.3.3 Algorithm

We now describe an algorithm called *Bi*nary feedback *A*nomaly *D*etector (*BiAD*) [83], that uses the recommendations made to the players and their feedback to decide between one of the two hypotheses. In every *round* of recommendation, each player is recommended an item by the recommendation engine. In round t, the algorithm uses the feedback from the players and computes for each item, the total number of players until that round who have been recommended that item and found the item ineffective. This number is denoted by $B_i(t)$ for item i. If the sum of the largest $\hat{A}(t)$ of these numbers among all the items is greater than or equal to a threshold $T(t)$, the recommendation engine is declared to be *biased*. Otherwise, the same procedure is repeated in the next round. If the algorithm does not declare the engine to be *biased* in $Q(m)$ rounds, then the hypothesis that the engine is *biased* is rejected. There are three parameters associated with the algorithm – $\hat{A}(t)$, $T(t)$ and $Q(m)$ – whose choice is governed by Equations 2.1-2.7). These parameters correspond to the following quantities:

- $\hat{A}(t)$: This parameter corresponds to the size of the subset of movies used to compute the quantity of interest $S(t)$ in round t. This has been elaborated in the definition of $S(t)$ in Algorithm 1. The choice of this parameter we use is given by Equation 2.1.

- $T(t)$: This is a time varying threshold in the algorithm which is compared with the quantity $S(t)$; the choice of this threshold has been elaborated in Theorem 2.1.

- $Q(m)$: This is the maximum number of rounds of recommendation for which the algorithm operates before declaring a recommender engine as *biased* or *objective*.

The pseudocode for this algorithm is shown in Algorithm 1.

Algorithm 1 *Bi*nary feedback *A*nomaly *D*etector (*BiAD*)

Initialize $t = 1$ (round 1).
while $t \leq Q(m)$ **do**
  Compute $B_i(t)$ = number of players who have rated item i ineffective upto round t for all $i \in [m]$.
  Compute $S(t) = \max_{\{\hat{\mathcal{A}} \subseteq [m] : |\hat{\mathcal{A}}| = \hat{A}(t)\}} \sum_{i \in \hat{\mathcal{A}}} B_i(t)$
  if $S(t) \geq T(t)$ **then**
    Stop and accept H_1.
  else
    $t \leftarrow t + 1$.
  end if
end while
Stop and reject H_1.

As opposed to the basic average test, this algorithm searches for concentration of large number of ineffective items in a small set. Since the number of potential advertisements is unknown, this algorithm makes decisions in real-time as it gets feedback from the players. Larger the size of the ad-pool, larger is the number of feedback samples required to detect a *biased* engine. (The trade off between various parameters is discussed in detail in Section 2.4.) Therefore, the algorithm increases the size of the search set with progressing rounds of recommendation. Also, note that the algorithm requires only binary feedback from the players – whether the recommendations are effective or ineffective, which explains the name of the algorithm.

The following theorem gives sufficient conditions for good performance of the algorithm. Unlike in general hypothesis testing problems, we define Type I error, which corresponds to false positives, only for *objective* systems. On the other hand, Type II error is used to refer to missed detection in the case of a *biased* system. We do not give any guarantees for the class of recommendation engines that are neither *objective* nor *biased*.

Theorem 2.1. *Let the parameters in the detection algorithm, BiAD satisfy the following equations:*

$$\hat{A}(t) = t, \tag{2.1}$$

$$= \min\left\{nt, \exp\left(1 + W\left(\frac{\hat{\beta}(t)}{e}\right)\right)\hat{p}(t)\right\}, \tag{2.2}$$

where $W(\cdot)$[2] *represents the Lambert-W or product log function, and*

$$\hat{\beta}(t) = \left(\frac{\left(\hat{A}(t) + c\right)\log m}{\hat{p}(t)} - 1\right)^{+}, \tag{2.3}$$

$$\hat{p}(t) = \exp\left(1 + W\left(\frac{\beta(t)}{e}\right)\right)p(t), \tag{2.4}$$

$$\beta(t) = \left(\frac{\left(\hat{A}(t) + c\right)\log m}{p(t)} - 1\right)^{+}, \tag{2.5}$$

$$p(t) = ntc' + \max_{\left\{\hat{\mathcal{A}}\subseteq[m]:|\hat{\mathcal{A}}|=\hat{A}(t)\right\}}\left\{\sum_{u=1}^{n}\sum_{l=1}^{t}\mathbb{E}\left[P_u^{\hat{\mathcal{A}}}(l)\right]\right\}, \tag{2.6}$$

$$P_u^{\hat{\mathcal{A}}}(l) = \sum_{i\in\hat{\mathcal{A}}}\frac{\mathbb{1}\left\{R_{ui} < \eta\right\}}{F\left(\hat{R}_{ui}(l), \hat{\mathbf{R}}_u(l)\right) - l + 1}, \tag{2.7}$$

where $(\cdot)^+ = \max(\cdot, 0)$*, and* c, c' *are system design parameters.*

[2]For any $z \in \mathbb{R}$, $W(z)e^{W(z)} = z$.

For $c = 1/2$, BiAD gives the following guarantees on the error probabilities:

(I) Type I Error:

If the recommendation engine is objective, and if

(a) the feedback errors to the anomaly detection system are such that $\{\xi_u,\ u \in [n]\}$ satisfy $\frac{1}{n}\sum_{u=1}^{n}\xi_u \leq c'$,

(b) the estimation errors $\boldsymbol{\epsilon}(t)$ of the recommendation engine, the number of rounds $Q(m)$ and c' are such that $\frac{p(t)}{nt} \leq \frac{1}{4}\,\forall\, 1 \leq t \leq Q(m)$, and

(c) the number of players $n = \omega(\log m)$,

then the probability that BiAD declares it to be anomalous is $O(\frac{Q(m)}{\sqrt{m}})$.

(II) Type II Error:

If the recommendation engine is anomalous with an ad-pool of size A, and if, along with Condition (I)a, the following conditions hold:

(a) the number of ads, $A \leq Q(m)$,

(b) the fraction of recommendations that are ads, i.e., the bias probability $\gamma \geq \left(\frac{e}{W(1)}\right)^2 \frac{1}{(1-\delta)(1-c')} \max\left(\frac{\log m}{n}, \frac{p(A)}{nA}\right)$ for some constant $\delta \in (0,1)$, and

(c) $\sum_{u=1}^{n} f_u(\eta, \mathcal{A}) = o(\gamma n A)$, where $f_u(\eta, \mathcal{A})$ is the number of effective ads for user u,

then the probability that BiAD does not declare the system as anomalous within A rounds is $e^{-\Omega((1-c')\gamma n)}$.

The proof of this theorem is presented in Appendix A.1.

2.4 Discussion

In this section, we discuss how the error probabilities depend on the parameters of the problem.

2.4.1 Choice of Threshold

Note that computation of the threshold function, $T(t)$ as specified in Theorem 2.1 (given by Equation (2.2)) requires knowledge of the noise statistics and also the players' opinions about all the items in the database. More precisely, since $\hat{\mathbf{R}}(t) = \mathbf{R} + \boldsymbol{\epsilon}(t)$, computation of $\mathbb{E}\left[P_u^{\hat{\mathcal{A}}}(l)\right]$ (see Equation (2.7)) requires knowledge of $\mathbf{R}_u$ and also the distribution of estimation noise, $\boldsymbol{\epsilon}_u(l)$. The noise statistics reflect the accuracy of the learning strategy of the recommendation engine, and it is possible that these statistics are unknown or cannot be estimated. Moreover, it might also be difficult to obtain the players' opinions about all the items in the database. To overcome this difficulty, a practical implementation of the algorithm could use an approximation of the unknown quantity. We now propose one way to compute such an approximation. Note that

$$\begin{aligned}\mathbb{E}\left[P_u^{\hat{\mathcal{A}}}(l)\right] &= \mathbb{E}\left[\sum_{i\in\hat{\mathcal{A}}} \frac{\mathbb{1}\left\{R_{ui} < \eta\right\}}{F\left(\hat{R}_{ui}(l), \hat{\mathbf{R}}_u(l)\right) - l + 1}\right] \\ &\leq \mathbb{E}\left[\sum_{i\in\hat{\mathcal{A}}} \frac{1}{F\Big(\eta + \epsilon_{ui}(l), \mathbf{R}_u(l) + \boldsymbol{\epsilon}_u(l)\Big) - l + 1}\right],\end{aligned}$$

where the inequality follows since $F\Big(r, \mathbf{R}_u(l) + \boldsymbol{\epsilon}_u(l)\Big)$ is a non-increasing function of r. We assume that the estimates of the ratings are not skewed in one direction, and

therefore the noise has zero mean. Since the noise statistics are unknown, we could approximate the right hand side of the above inequality by substituting the noise term with its mean. With this approximation, the right hand side of the inequality can be substituted with

$$\sum_{i \in \hat{\mathcal{A}}} \frac{1}{F\Big(\eta, \mathbf{R}_u(l)\Big) - l + 1} = \frac{\hat{A}(t)}{f_u(\eta, [m]) - l + 1}, \tag{2.8}$$

where $f_u(\eta, [m])$ is the total number of effective items in the database for user u. Depending on the application, it might be relatively easy to estimate this number or at least estimate a lower bound for this number. As an example, one could roughly estimate that for every user there are $\sqrt{m}$ effective items among the m items in the database. We observe in our simulations that a rough estimate of $f_u(\eta, [m])$ is sufficient to obtain good results.

Note that over-estimation (under-estimation) of $T(t)$ decreases the probability of Type I (Type II) error and increases the probability of Type II (Type I) error. In other words, the higher the value of $T(t)$, the lower is the probability of Type I error and the higher is the probability of Type II error. Therefore, the risk associated with false positives and missed detection could serve as a guideline for the choice of the threshold function. In our simulations (Section 2.5), we propose a practical threshold function that gives a good balance between the two error probabilities for most scenarios.

2.4.2 Effect of Parameters on Performance

Theorem 2.1 gives guarantees on the asymptotic performance of $BiAD$ as the size of the database grows large. These guarantees depend on various parameters

in the algorithm as well as the recommendation engine. From the analytic bounds derived in Theorem 2.1, we analyze in this section the trade off between these parameters to understand the conditions under which the algorithm shows good performance. We see that the theoretical results support our intuitive understanding about the conditions under which a *biased* system can be distinguished from an *objective* system. These results are also corroborated by our simulation results described in Section 2.5.

In Section 2.4.1, we consider the effect of the choice of the threshold parameter on the error probabilities. We now discuss the effect of other parameters.

Number of Rounds in the Test and Size of the Ad-Pool. It is seen (from Result (I)) that the upper bound on the probability of Type I error increases with increasing number of rounds. This is expected, since it gives more chances to falsely declare a system *biased*. For Type I error to go to zero as the size of the database goes to infinity, it is sufficient if $Q(m) = o(\sqrt{m})$.

Guarantees for detection of a *biased* engine (Result (II)) are dependent on various parameters. One of the conditions is that the ad-pool is not very large (Condition (II)a). Specifically, it is sufficient if the size of the ad-pool, A is at most the maximum number of rounds of recommendations, $Q(m)$. Therefore, increasing $Q(m)$ (the number of rounds of testing) enables detection of larger ad-pools but also increases the probability of Type I error. Intuitively, a small ad-pool conforms with our definition of a *biased* recommendation engine as one that favors a few items over many others and therefore facilitates easier detection.

Number of Effective Ads. For correct detection of a *biased* engine, it is also required that the average number of effective ads (averaged over all players) is not very large (Condition (II)c). A large number of effective ads enables the recommendation system to customize ads according to users' tastes and is contradictory to our interpretation of a *biased* system which recommends ads that do not match with users' preferences.

Number of Players. The dependence on the number of players n is seen in two respects – it determines the minimum bias probability at which detection is guaranteed (Condition (II)b) and also the probability of Type II error. Both these results show that a large number of players improves the prospect of correct identification which can explained by the fact that a large sample size supports better statistical analysis.

Number of Effective Items. The minimum bias probability at which detection is ensured is also determined by the average number of effective items in the entire database. This can be seen from the term $\frac{p(A)}{nA}$ in (Condition (II)b). The no estimation noise case ($\epsilon_{ui}(t) = 0$ for all u, i, t) is useful in understanding the term $\frac{p(A)}{nA}$. When there is no noise, $\frac{p(A)}{nA} = O\left(\frac{1}{nA}\sum_{u=1}^{n}\sum_{l=1}^{A}\frac{A}{f_u(\eta,[m])-l+1}\right)$. Therefore, $\frac{p(A)}{nA}$ has an inverse relation with the number of effective items in the database. This conveys that a large number of effective items facilitates better detection of a *biased* engine. Intuitively, a large number of effective items in the database helps in clearer demarcation of an *objective* engine from a *biased* one. With many effective items in the database, an *objective* system would have a higher probability of recommend-

ing effective items, while a *biased* system always makes at least γ fraction of its recommendations from the ad pool where the number of effective ads is limited.

Bias Probability. The more a *biased* engine recommends from the ad-pool, the more apparent is its biased behavior. The fraction of total recommendations that are from the ad-pool is captured by the bias probability, γ. We see that the probability of Type II error decays exponentially with increasing γ, and also that larger γ facilitates easier anomaly detection (Conditions (II)b, (II)c).

Choice of c, c'. In the course of the proof of Theorem 2.1, we prove that for any choice of c, the Type I Error is bounded by $O(\, Q(m)\, m^{-c}\,)$. Hence, by changing c in the algorithm, one can control the error probability. However, the downside of increasing c is that it effectively increases the threshold $T(t)$, which results in requiring the bias probability $\gamma = \Omega(\log m(1 + (c/A))/n)$.

Likewise, the choice of c' determines the tradeoff between the algorithm's tolerance for errors in feedback and its ability to correctly identify a *biased* engine. This tradeoff can be observed from the conditions given in Theorem 2.1. The higher the value of c', the greater is the range of feedback errors for which the guarantees hold (Condition (I)a). However, this results in poorer guarantees for detection of bias reflected by Conditions (I)b, (II)b and higher Type II Error probability.

2.4.3 Applications

The proposed anomaly detection algorithm is readily applicable in the retail market. It can identify recommender systems that dole out sponsored advertise-

ments in the garb of personalized recommendations. In this era of personalization, there are numerous other applications, two of which are described below. These two examples also illustrate the advantage of *BiAD* in requiring simple binary feedback, allowing it to be applied in a wide variety of scenarios.

Search Engine Bias. Search engine bias is one of the most important ethical issues surrounding search engines, and its social implications have been studied for more than a decade [71, 148, 50]. There has been increasing awareness about the influence of search results ranking on election results in democracies. In [50] experiments performed with 1800 undecided voters in India, revealed that alteration of search results can create a shift of 12.5 % in voting decisions. The Stanford Encyclopedia of Philosophy [131] describes search engine bias as non-neutrality of search engines, where "search algorithms do not use objective criteria" or "favor some values/sites over others in generating their list of results for search queries." A sponsored search engine in the late 1990s called GoTo ranked its search results purely based on bids from advertisers [48]. It was evidently unsuccessful due to users' mistrust of paid searches and was eventually acquired by Google. Google also uses an auction to sell ads but displays them physically separated from organic search results.

The pros and cons of enforcing transparency in the algorithms used for generating search results have been examined in [48, 62]. Even in the absence of total transparency, anomaly detection systems such as *BiAD* could be useful in identifying bias in search engines. With personalization being extended to search results [45, 69, 43], search engines virtually act as recommendation engines. With large number of potential search results, our model of recommender systems with a large

database fits well in this problem where biased search engines correspond to *biased* recommender systems. To be more precise, personalized search result ranking algorithms like PageRank [69] can be viewed as the learning strategy of the recommendation system, while alteration of search results due to malicious reasons can be modeled as bias in the recommendation engine. Feedback from users can be obtained through their clicks on recommended links which implicitly indicate whether or not the search results were effective. In addition to search engines, this example can be extended to identify hidden sponsored advertisements in social networking sites and online news portals, all of which use personalization algorithms.

Pharmaceutical Lobby. Pharmaceutical lobbying is another controversial issue that affects many parts of the world [10, 93, 46, 125]. Among its many aspects, we focus on the marketing practices of large pharmaceutical companies which manipulate the opinions of doctors, health care providers and law-makers by providing biased information and through other tactics [29, 52]. There have been allegations that big drug companies influence physicians to prescribe their highly priced branded drugs even when other better or cheaper alternatives are available [92, 59].

Again, our interpretation of a *biased* recommendation engine is well-suited to model this scenario. Since drugs are prescribed on a person-to-person basis, health care providers can be viewed as recommendation service providers who recommend drugs to patients, and the lobbying drug companies act as advertisers. A health care system that favors a few incompetent (expensive or ineffective) drugs in spite of the availability of other *better* (cheaper or more effective) alternatives matches well with our definition of a *biased* recommendation engine. With data samples consisting of

prescriptions and their efficacy on patients, anomaly detection algorithms like *BiAD* could help watchdog agencies in identifying such malpractices.

2.5 Numerical Results

We evaluate our algorithm through offline simulations, with careful considerations for ensuring proximity to real world scenarios.

2.5.1 Simulation Setup

Given below is a detailed description of the methods we adopt in our simulations to replicate the different components of a recommender system.

User-Item Database. Estimating users' opinions about all items in a database is essential for real-world recommender systems, and hence for simulating those recommender systems as well. However, the ground truth on such data set is not available, since in existing data sets each user typically only rates a small subset of items, and those ratings are also noisy and possibly biased.

For a complete user-item rating matrix, we take available sparse data sets ([104, 25], *MovieLens*) and renormalize the ratings on a linear scale from zero to ten. The missing entries in the sparse matrix are then filled in using the matrix completion algorithm from [78]. For the purpose of simulating real-world user opinions, we consider this completed matrix as ground truth. We evaluate our algorithm on three data sets:

D1 a subset of the Amazon cellphones and accessories data set [104] with 3671 users and 8728 items,

D2 a subset of the Netflix Prize data set [25] with 2951 users and 9259 movies, and

D3 a subset of the *MovieLens 10M* ratings data set with 3671 users and 8729 movies.

Recommendation Engine. Due to non-transparency of recommendation strategies, it is not exactly known how recommendation engines behave. As a representation of the learning strategies used by these systems, we use two learning algorithms popular in literature:

L1 Matrix factorization. Specifically, we use the inexact ALM method proposed in [98].

L2 User-based collaborative filtering (with Pearson correlation as the similarity metric [117]).

To simulate the temporal dynamics of a recommender system, the recommendation engine is initially supplied a sparse subset of the user-item ratings chosen according to a power-law degree distribution observed in real-world data sets [70]. (Specifically, the number of feedback entries from each user is chosen from a $pareto(3, 3)$ distribution.) In each round of recommendation, the engine recommends one item to each user and observes users' feedback about the recommended items. It periodically updates its estimate of the users' preferences based on this feedback. In our experiments, we set the frequency of these updates to once in every 5 rounds.

It is natural for a recommendation engine to have an *explore* component

to address the cold start problem and have wider coverage of the database [95]. Therefore, in all our experiments we invoke random explore for 0.1 fraction of the recommendations made. In a recommendation meant for exploration, an item is chosen uniformly at random from the database, provided it has not been shown previously.

For all other recommendations which do not explore, we use the following recommendation strategy – for each user, the items are ranked according to the estimated preferences. To make a recommendation, an *objective* recommendation engine chooses the highest ranked item among the items not yet recommended. The recommendation strategy of a *biased* engine follows the description in Section 2.2.2.2 – for any recommendation, with probability $1-\gamma$, like an *objective* engine, it recommends the highest ranked item and with probability γ, it recommends an item from the ad-pool. We consider two kinds of ad selection strategies – from the among the ads that have not already been recommended to the user,

A1 An ad item is chosen uniformly at random.

A2 The ad item which has the highest ranking is chosen.

Strategy A2 corresponds to customization of ads according to users' tastes. Note that this is harder to detect than strategy A1 since it has higher likelihood of recommending effective ads.

Anomaly Detection System. For players in the anomaly detection system, we randomly choose a subset of users from the data set. In experiments which test the performance of the algorithm with increasing number of players, we choose the

subset of players incrementally.

As explained in Sections 3.3 and 2.3, the algorithm requires feedback samples of efficacy of the recommendations made to the players. For our experiments, we adopt the characterization of efficacy used in our theoretical model given by Definition 2.1. Note that the number of effective items in the database for any user depends on the efficacy threshold η. To be able to test the algorithm for different number of effective items, we choose a different efficacy threshold for each of the data sets. Specifically, we set $\eta = 5.5, 8.0$, and 8.8 for data sets D1, D2, and D3, which correspond to an average number of effective items of 80, 250, and 150 respectively.

2.5.2 Results

We evaluate the performance of *BiAD* with variations in different parameters of the recommender system and the anomaly detection system. To demonstrate its effectiveness in different settings, we present performance results for various combinations of data sets (D1-D3) and recommendation algorithms (L1-L2, A1-A2). An *objective* recommendation engine is represented by its learning algorithm (L1 or L2) while a *biased* recommendation engine is represented by its learning algorithm (L1 or L2) and its ad-recommendation strategy (A1 or A2). Although we present experimental results for specific combinations for space limitations, other settings give similar trade offs. Specifically, the simulation results corroborate our theoretical analysis of the tradeoffs between various parameters in Section 2.4.

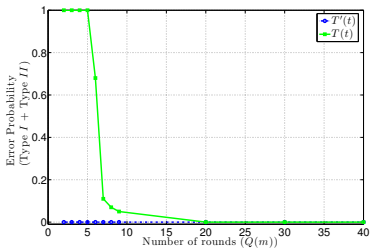
We now describe how the performance depends on the choice of various parameters in *BiAD*. We set the parameter $\hat{A}(t)$ according to Equation (2.1) in all

the simulations. Other parameters of the algorithm are discussed below.

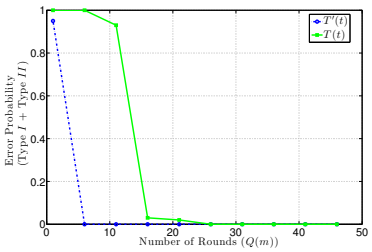
Threshold. As explained in Section 2.4.1, varying the threshold parameter $T(t)$ in the algorithm affects Type I and Type II error probabilities in opposite ways. Using lower values of threshold increases probability of Type I error and decreases that of Type II error. The threshold given by Equation (2.2) is designed to ensure, irrespective of the number of players in the anomaly detection system, low probability of false positive (Type I error) even when the estimated user preferences are noisy. This is especially important if the risk associated with false implication of an *objective* recommendation engine is high.

We observe that a less conservative threshold gives a better balance between the two types of errors. Specifically, we use a threshold that can be proved to guarantee low error rates under the assumption that the recommendation engine's estimation of user preferences are accurate. This threshold, denoted by $T'(t)$, is equal to the value of $\hat{p}(t)$ given by Equation (2.4). In all our simulations, we show the performance of $BiAD$ for both these threshold choices. Simulation results show that $T'(t)$ gives better performance than $T(t)$ except in one case (Figure. 2.2b) where those two choices give similar performances.

In both these thresholds, $\mathbb{E}[P_u^{\hat{A}}(l)]$ in Equation (2.6) is substituted with the right hand side of (2.8). This requires knowledge of the number of effective items for each player in the anomaly detection system. In our simulations, $BiAD$ approximates this with the average number of effective items for all the users. Effectively,



(a) Data set : D1 , Algorithm : L2 + A1 , $n = 100, \gamma = 0.45, A = 8$.



(b) Data set : D2 , Algorithm : L2 + A2 , $n = 100, \gamma = 0.35, A = 8$.

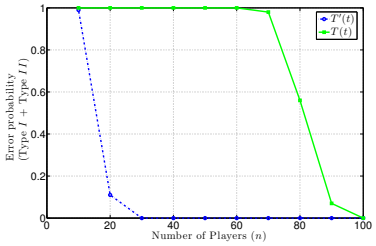
Figure 2.1: Number of rounds in the test, $Q(m)$ affects the number of ads that can be detected – at least 8 and 15 rounds required for $T'(t)$ and $T(t)$ respectively.

it uses the following value of $p(t)$ instead of Equation (2.6):

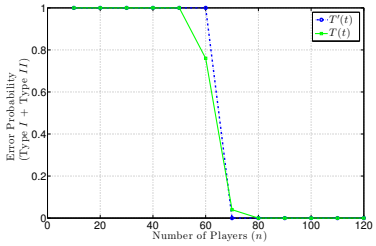
$$p(t) = \sum_{l=1}^{t} \frac{n\hat{A}(t)}{\tilde{f}([m]) - l + 1}, \tag{2.9}$$

where $\tilde{f}([m])$ is an estimate of the average number of effective items in the database $[m]$. We assume that this average number is not very difficult to estimate and in all the results, unless specified, *BiAD* has an accurate estimate of this number.

Number of Rounds in the Test. The number of rounds of recommendation $Q(m)$ affects the error probability. This is seen in Figure 2.1 which shows the variation of sum of Type I and Type II errors with $Q(m)$. Type I error rate is in fact close to zero for all values of $Q(m)$ for both the thresholds, so the plots effectively show Type II error rates. Theorem 2.1 guarantees detection of a *biased* engine if $Q(m) \geq A$. The plots show that *BiAD* detects 8 ads if $Q(m)$ is at least 8 and 15 for $T'(t)$ and $T(t)$ respectively. For all the remaining simulations, we set the parameter $Q(m) = 40$.



(a) Data set : D1, Algorithm : L1 + A1, $A = 8, \gamma = 0.45$.



(b) Data set : D2, Algorithm : L1 + A1 , $A = 8, \gamma = 0.35$.

Figure 2.2: The performance improves with number of collaborating users n.

Number of Players. Larger number of players in the anomaly detection system indicates higher number of input samples to the algorithm, and as expected, the algorithm performs better as this number increases. In Figure 2.2, we plot the sum of Type I and Type II error rates with increasing number of users. To detect a *biased* engine with the specified value of γ, these plots show that 70 and 100 players respectively are sufficient when $T'(t)$ and $T(t)$ are chosen to be the threshold parameter. We use 100 players in all other simulations.

In addition to the choice of parameters in the algorithm, various aspects of the recommender system affect the performance of *BiAD*. These are described below.

Size of the Database. Theorem 2.1 shows that *BiAD* performs well for recommender systems with large item databases. Databases of varying size are constructed by sub-sampling items from the original data set. Figure 2.3 shows the variation of Type I and Type II errors with the size of the database. $T(t)$ and $T'(t)$ have very

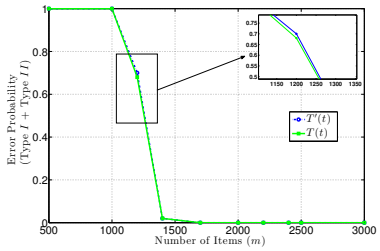


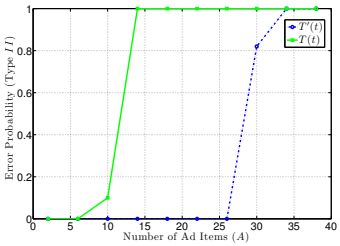
Figure 2.3: Variation of Type I + Type II error rates with size of data set. Data set : D2, Algorithm : L1 + A2, $A = 8, \gamma = 0.35, n = 100$.. When there are more choices to recommend, the user satisfaction with *objective* recommender systems improves making detection easier.

similar performance for the parameters in this experiment. The plot shows that, for detection of 8 ads recommended 35 percent of the time, the algorithm is effective for databases of size 1500 items or larger.

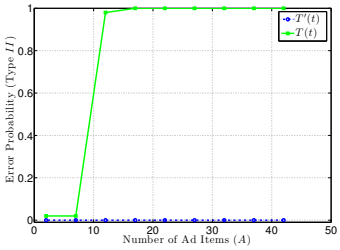
We now demonstrate that *BiAD* has been appropriately designed to identify *biased* engines that systematically recommend (make a sizable fraction of recommendations) from a small ad-pool.

Size of the Ad-Pool. Theorem 2.1 shows that *BiAD* guarantees detection of a *biased* engine that has a small ad-pool. This same effect is also observed in simulations – Figure 2.4 shows rate of missed detection (Type II error rate) with varying size of the ad-pool. It is seen that both the thresholds perform well for small number of ads, while threshold $T'(t)$ can detect an ad-pool of size upto 25.

Bias Probability. The bias probability γ quantifies the intensity of bias of the recommendation engine. Plots (Figure 2.5) for Type II error rate with γ show that *more biased* (higher γ) engines are easier to detect.

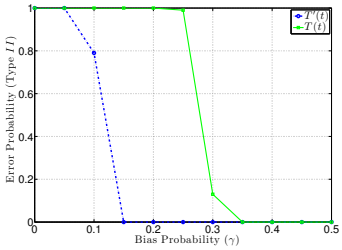


(a) Data set : D1 , Algorithm : L1 + A1 , $n = 100, \gamma = 0.45$.

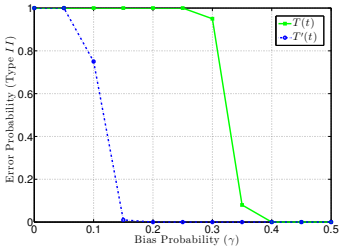


(b) Data set : D2 , Algorithm : L2 + A2 , $n = 100, \gamma = 0.35$.

Figure 2.4: As the size of the ad-pool A increases, the (personalized) ads become similar to effective recommendations, making it hard to detect (Type II error is large).



(a) Data set : D2 , Algorithm : L1 + A2 , $n = 100, A = 8$.



(b) Data set : D3 , Algorithm : L1 + A2 , $n = 100, A = 8$.

Figure 2.5: Type II error rate decreases as bias probability γ increases.

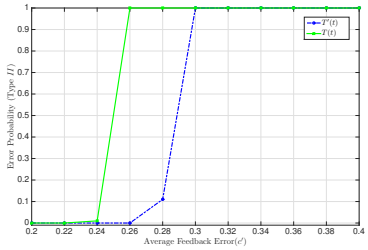


Figure 2.6: Variation of Type II error rates with feedback error probability (c'). Data set : D2, Algorithm : L1 + A1, $A = 8, \gamma = 0.5, n = 2000, Q(m) = 10$. Error in detection increases with the increase in feedback error

Average Feedback Error(c'). Theorem 2.1 shows that the probability of detecting a *biased* recommender system goes down with increase in feedback error c'. This can be observed in Figure 2.6 where Type II error is plotted against increasing c' while all other parameters are held constant. In Figure 2.7 Type II error is plotted against bias probability (γ) in the presence of feedback error of 10%. With 1000 players, the threshold $T'(t)$ has zero error in the presence feedback error for bias probability as low as 0.2.

With increasing feedback error, higher number of players are required to obtain performance comparable to the setting when there is no feedback error. Therefore, in practice, the effect of feedback errors can be mitigated by having sufficient number of participating players in the system. Note that, for binary feedback, an error implies a complete flip in feedback and therefore, even small percentage of errors can be significant. The simulations above demonstrate the robustness of BiAD against feedback errors.

Estimate of the Number of Effective Items. In all the simulations above, it is

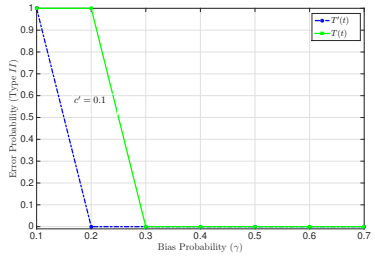


Figure 2.7: Variation of Type II error versus Bias probability (γ) in the presence of feedback error. Data set : D2, Algorithm : L1 + A1, $A = 8, n = 1000, Q(m) = 10$. Performance of detection strategy decreases with an increase in feedback error rate.

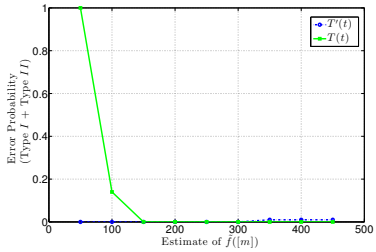


Figure 2.8: Variation of Type I + Type II error rates with perturbations in the algorithm's estimate of the average number of effective items $\tilde{f}([m])$. Data set : D3, Algorithm : L1 + A1, $A = 8, \gamma = 0.4, n = 100$.

assumed that *BiAD* has an accurate estimate of the average number of effective items ($\tilde{f}([m])$) which is used to determine the threshold parameter in the algorithm (See Equation (2.9)). Note that overestimation of this parameter lowers the threshold parameter thereby increasing the probability of Type I error and decreasing the probability of Type II error. Figure 2.8 shows the effect of variations in this estimate for data set D3 which has an average of 150 effective items. We observe that $T'(t)$ performs well for a wide range of estimates. In the case of $T(t)$, it is safer to overestimate the parameter $\tilde{f}([m])$ than to underestimate it.

2.5.3 Ineffectiveness of Basic Average Test

As explained in the introduction (Section 4.1), we demonstrate the inability of the basic average test to distinguish between random errors and deliberate promotion of ads. This test computes the average rating across all recommendations and decides between the two hypotheses based on a threshold parameter. With the specifics of the recommendation strategy (explore probability) unknown, it is difficult to estimate the right value of threshold. For an explore probability of 0.1, Figure 2.9 shows the performance of the basic average test for different values of the threshold, denoted τ. It is seen that threshold values around 3 give the best performance. But, as shown in Figure 2.10, this same threshold value fails for other values of explore probability. For example, the basic average test falsely declares an *objective* recommendation engine with 20 percent explore probability as *biased*. This shows that the correct choice of τ is sensitive to the explore probability. In contrast, note that *BiAD* has nearly zero Type I error rate for all values of explore probabilities.

2.6 Summary

We propose an algorithm that can identify an *biased* recommendation engine that systematically favors a few sponsored advertisements over other genuine recommendations. We formulate a probabilistic model for recommender systems and give theoretical guarantees for our detection algorithm based on this model. Specifically, we show that the probability of missed detection and false positives are low for recommender systems with large databases. We show through simula-

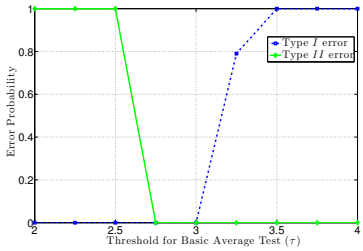


Figure 2.9: Variation of Type I and Type II error rates with threshold τ for the basic average test shows that the naive approach is sensitive to the choice of parameter τ. Data set : D1, Algorithm : L1 + A1, $A = 8, \gamma = 0.45, n = 100$, Explore Probability = 0.1.

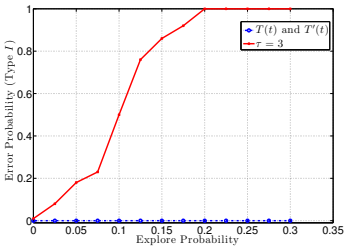


Figure 2.10: Variation of Type I error rates with variation in explore probability shows that the threshold for the basic average test is sensitive the value of explore probability. Data set : D1, Algorithm : L1, $n = 100$.

tions that the algorithm performs well for many data sets and different types of recommendation algorithms. In an age when both personalization and advertising have become very prevalent, this kind of anomaly detection algorithm is relevant in a wide variety of scenarios. We demonstrate how our detection algorithm can be applied to problems such as identification of search engine bias and pharmaceutical lobbying. It would be interesting to investigate ways of deploying such an anomaly detection mechanism in practical settings.

Chapter 3

Regret of Queueing Bandits

3.1 Introduction

A range of queueing systems face the following basic challenge: scheduling decisions must be made to optimize a desired performance criterion (e.g., small queue lengths, low delay, etc.); but relevant system parameters (such as arrival rates and service rates) may be imperfectly known. These issues are particularly relevant for scheduling in *dynamic* environments. In such systems, compatibility between different types of jobs and servers may change over time; servers may come and go over time; and the arrival and service rates themselves may shift over time[1].

For example, consider *scheduling in wireless systems.* Suppose a given user can transmit over any one of K channels; which channel(s) should we use to serve the user? In general, user-channel performance will be imperfectly known and change over time. Therefore a scheduling policy must learn and adapt to these changes in finite time. Similarly, consider *task allocation in crowdsourcing systems.* Suppose a given type of task can be assigned to any one of K types of workers on an ongoing basis; which worker is the best fit for this type of task? In general, we will need to learn the skill compatibility of a worker for a given type of task. Since workers may

[1]Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, and Sanjay Shakkottai. "Regret of queueing bandits". In Advances in Neural Information Processing Systems 29, pages 1669–1677. 2016. Co-authors of the paper made equal contributions in obtaining these results.

come and go in such a system, again compatibility must be learned reasonably well in finite time.

Traditionally, scheduling algorithms have focused on delivering optimal performance, given a known operating environment; this is *exploitation*. By contrast, our work focuses on the additional requirement introduced by environments where system parameters are not perfectly known: the scheduling algorithm must include *exploration* to learn about these parameters. The seminal dynamic optimization problem that isolates the exploration-exploitation tradeoff is the *multiarmed bandit* (MAB) problem. In our we study a variant of an MAB problem that is appropriate for scheduling in queueing systems.

Formally, we consider a system that is a discrete-time stochastic switched network with U queues (one for each job type) and K servers ($U \leq K$). Arrivals to the queues, and services offered by each server to each queue, are i.i.d. across time slots, according to a product Bernoulli distribution. In principle any queue can be served by any server; but the service rates are heterogeneous across queue-server pairs (also referred to as *links*). We assume the scheduler is uninformed about the link service rates, and in addition may be uninformed about the arrival rates as well.

In any time slot, each server can be assigned to at most one queue and each queue can be assigned to at most one server; thus in each time slot, scheduling amounts to choosing a matching in the complete bipartite graph between queues and servers. This problem clearly has the explore-exploit tradeoff inherent in the MAB problem: since the service capacities across different servers are unknown to the scheduler, there is a tradeoff between learning (*exploring*) the capacities of different

links and scheduling (*exploiting*) the most promising server from past observations. We refer to this problem as the *queueing bandit*.

To focus our analysis on this tradeoff, we study a special case of the scheduling problem with two additional assumptions. *First*, we assume there is single unique matching which is strictly better than all other matchings for all queues (called the *optimal matching*). *Second*, we assume the arrival rates are within the capacity region; i.e., if the optimal matching is chosen, the system is stable. An unique optimal matching means that we are solving a pure coordination problem between queues and servers (e.g. unique worker bested suited for a job type in an online crowdsourcing system). With these assumptions, our objective is to design a joint scheduling and learning algorithm that learns and uses the optimal matching efficiently.

In this work, we define the *cost* for this problem as the U-dimensional vector of queue lengths. Let $\mathbf{Q}(t)$ be the queue length vector at time t under a given scheduling policy, and $\mathbf{Q}^*(t)$ be the corresponding vector under the "genie" policy that always schedules the optimal matching. We define *queue-regret* as the difference in mean queue lengths for the two policies. That is, the regret (vector) is given by

$$\mathbf{\Psi}(t) := \mathbb{E}\left[\mathbf{Q}(t) - \mathbf{Q}^*(t)\right]. \tag{3.1}$$

We refer to the regret vector $\mathbf{\Psi}(t)$ as the *queue-regret*; formally, our goal is to develop algorithms that minimize the queue-regret.

To develop some intuition, we start by comparing this to the standard stochastic MAB problem. In a basic formulation of the MAB problem (see, e.g.,

[32]), there are K arms, each with an unknown reward distribution. At each time step the algorithm can choose one of these arms; the goal is to minimize the expected *cumulative* regret against the genie policy that knows the best arm. Well-known algorithms such as UCB, KL-UCB, and Thompson sampling achieve a regret of $O((K-1)\log t)$ at time t [21, 55, 13]. This result is essentially tight, in the sense that there exists a lower bound of $\Omega((K-1)\log t)$ over all policies in a reasonable class, so-called α-consistent policies [91].

Now note that, in a queueing system with a single queue and K servers, we can obtain a simple bound on the queue-regret by observing that it cannot be any higher than the cumulative regret over scheduled services. Thus we immediately obtain an upper bound of $O((K-1)\log t)$ for the queue regret using standard bandit algorithms like UCB, Thompson Sampling etc. However, this upper bound does not tell the whole story for the queueing bandit.

3.1.1 Contribution of this Thesis

We show that there are two "stages" to the regret behavior in queueing bandit. In the *early* stage, the bandit algorithm is unable to even stabilize the system; therefore in this region, the queue-regret grows with time, similar to the cumulative regret. Once the algorithm is able to stabilize the queue—the *late* stage, a dramatic shift occurs in the behavior of the queue regret. Because the queue now goes through regenerative cycles (busy periods), the sample-path queue-regret essentially "resets" on regular intervals; i.e., the sample-path queue-regret becomes zero or below zero at these time instants. Thus the queue-regret should fall over

time, as the algorithm learns.

Our main results provide lower bounds on queue-regret for both the early and late stages, as well as UCB-inspired algorithms that essentially match these lower bounds. We first describe the late stage, and then describe the early stage for a heavily loaded system.

1. *The late stage.* We first consider what happens to the queue regret as $t \to \infty$. As noted above, a reasonable intuition for this regime comes from considering a bandit algorithm, but where the sample-path queue-regret "resets" at time points of regeneration[2]. Therefore, queue-regret accumulates only over regeneration cycles and not over the entire period from the beginning. In a sense, the queue-regret in this case can be considered as a (discrete) *derivative* of the cumulative regret. Since the optimal cumulative regret scales like $\log t$, asymptotically the optimal queue-regret should scale like $1/t$.

   Indeed, we show that the queue-regret for α-consistent policies is at least C/t infinitely often, where C is a constant independent of t. Further, we introduce an algorithm called Q-UCB for the queueing bandit, and show an asymptotic regret upper bound of $O\left(\text{poly}(\log t)/t\right)$ for Q-UCB, thus matching the lower bound up to poly-logarithmic factors in t. Q-UCB uses *structured exploration*: it exploits the fact that the queue regenerates regularly to explore more systematically and aggressively.

[2]When $\mathbf{Q}(t) = \mathbf{0}$, $\mathbf{Q}(t) - \mathbf{Q}^*(t)$ is non-positive – we refer to these time instants as regeneration points.

2. *The early stage.* The preceding discussion might suggest that an algorithm that explores aggressively would dominate any algorithm that balances exploration and exploitation. However, this intuition would be incorrect, because an overly aggressive exploration policy will preclude the queueing system from ever stabilizing, which is *necessary* to induce the regenerative cycles that lead the system to the late stage. As a simple example, it is well known that if the only goal is to identify the best out of two servers as fast as possible, the optimal algorithm is a balanced randomized experiment (with half the trials on one server, and half on the other) [19]. But such an algorithm has positive probability of failing to stabilize the queue, and so the queue-regret will grow over time.

   To even enter the late stage, therefore, we need an algorithm that exploits enough to actually stabilize the queue. We refer to the early stage of the system, as noted above, as the period before the algorithm has learned to stabilize the queues. For a heavily loaded system, where arrival rates approach the service rates of the optimal matching, we show a lower bound of $\Omega(\log t/\log\log t)$ on the queue-regret in the early stage. Thus up to a $\log\log t$ factor, the early stage regret behaves similarly to the cumulative regret (which scales like $\log t$). Therefore in this region a standard bandit algorithm would be essentially optimal. (Straightforward arguments show that because Q-UCB explores more aggressively, it incurs regret $O(\log^3 t)$ in the early stage.)

   Perhaps more importantly, our analysis shows that the time to switch from the early stage to the late stage scales at least as $t = \Omega(K/\epsilon)$, where ϵ is the

gap between the arrival rate and the service rate of the optimal matching[3]. In particular, we show that the early stage lower bound of $\Omega(\log t/\log\log t)$ is valid up to $t = O(K/\epsilon)$; on the other hand, we also show that, in the heavy-load limit, depending on the relative scaling between K and ϵ, the regret of Q-UCB scales like $O\left(\text{poly}(\log t)/\epsilon^2 t\right)$ for times that are arbitrarily close to $\Omega(K/\epsilon)$. In other words, Q-UCB is nearly optimal in the time it takes to "switch" from the early stage to the late stage.

3.2 Related work

Our work is related to several threads of the literature, as we briefly detail below.

MAB algorithms. Stochastic MAB models have been widely used in the past as a paradigm for various sequential decision making problems in industrial manufacturing, communication networks, clinical trials, online advertising and web-page optimization, and other domains requiring resource allocation and scheduling; see for e.g., [58, 101, 32].

The MAB problem has been studied in two variants, based on different notions of optimality. One considers mean accumulated loss of rewards, often called *regret*, as compared to a genie policy that always chooses the best arm. Most effort in this direction is focused on getting the best regret bounds possible at any *finite time* in addition to designing computationally feasible algorithms [32]. The other line of research models the bandit problem as a Markov decision process (MDP),

[3] $\epsilon \to 0$ in the heavy-load setting.

with the goal of optimizing *infinite horizon* discounted or average reward. The aim is to characterize the structure of the optimal policy [101]. Since these policies deal with optimality with respect to infinite horizon costs, unlike the former body of research, they give steady-state and not finite-time guarantees. Our work uses the regret minimization framework to study the queueing bandit problem.

Bandits for queues. There is body of literature on the application of bandit models to queueing and scheduling systems [101, 110, 72, 42, 35, 136, 109, 100]. These queueing studies focus on infinite-horizon costs (i.e., statistically steady-state behavior, where the focus typically is on conditions for optimality of index policies); further, the models do not typically consider user-dependent server statistics. Our focus here is different: algorithms and analysis to optimize finite time regret.

Switch scheduling. Switch scheduling has received a great deal of attention in the last two decades; see for e.g., [128] for a survey. Notably, many of the scheduling algorithms (e.g., queue-length-based backpressure scheduling algorithms) can yield near-optimal performance in the absence of information about arrival rates; but these algorithms still require information about server availability and capacity. By contrast, our work focuses on learning about all unknown aspects of the environment, as needed to deliver small overall queue lengths. Finally, the problem of identifying the right matchings in a bipartite graph has been formulated as a special case of the combinatorial/linear bandit problem [53, 37], but with a generic reward structure and not in the context of queue scheduling as in our case.

3.3 Problem Setting

We consider a discrete-time stochastic switch network with U queues (i.e., users) and K servers, where $U \leq K$. The queues and servers are indexed by $u = 1, \ldots, U$ and $k = 1, \ldots, K$ respectively. Arrivals to queues and service offered by the links are according to product Bernoulli distribution and i.i.d. across time slots. The mean arrival rates are given by the vector $\boldsymbol{\lambda} = (\lambda_u)_{u\in[U]}$ and the mean service rates by the matrix $\boldsymbol{\mu} = [\mu_{uk}]_{u\in[U],k\in[K]}$.

In any time slot, each server can serve at most one queue and each queue can be served by at most one server. The problem is to schedule, in every time slot, a matching in the complete bipartite graph between queues and servers. The scheduling decision at any time t is based on past observations corresponding to the services obtained for the scheduled matchings until time $t-1$. Statistical parameters corresponding to the service distributions are considered unknown.

The queueing system evolution can be described as follows. Let $\kappa_u(t)$ denote the server that is assigned to queue u at time t. Therefore, the vector $\boldsymbol{\kappa}(t) = (\kappa_u(t)_{u\in[U]})$ gives the matching scheduled at time t. Let $R_{uk}(t)$ be the service offered to queue u by server k and $S_u(t)$ denote the service offered to queue u by server $\kappa_u(t)$ at time t. If $\mathbf{A}(t)$ is the (binary) arrival vector at time t, then the queue-length vector at time t is given by:

$$\mathbf{Q}(t) = \left(\mathbf{Q}(t-1) + \mathbf{A}(t) - \mathbf{S}(t)\right)^+ .$$

Notation: Important notation for the problem setting can be found in Table 4.1. Boldface letters are used to denote vectors or matrices and the corresponding

non-bold letters to denote their individual components. Also, the notation $\mathbb{1}\{\cdot\}$ is used to denote the indicator function.

Regret Against a Unique Optimal Matching

We focus on a simple special case of the above switch scheduling system. In particular, we assume for every queue, there is a unique optimal server with the maximum expected service rate for that queue. Further, we assume that the optimal queue-server pairs form a matching in the complete bipartite graph between queues and servers, that we call the *optimal matching*; and that this optimal matching stabilizes every queue.

Formally, make the following definitions:

$$\mu_u^* := \max_{k\in[K]} \mu_{uk}, \quad u \in [U]; \tag{3.2}$$

$$k_u^* := \arg\max_{k\in[K]} \mu_{uk}, \quad u \in [U]; \tag{3.3}$$

$$\epsilon_u := \mu_u^* - \lambda_u, \quad u \in [U]; \tag{3.4}$$

$$\Delta_{uk} := \mu_u^* - \mu_{uk}, \quad u \in [U], k \in [K]; \tag{3.5}$$

$$\Delta := \min_{u\in[U],k\notin k_u^*} \Delta_{uk}; \tag{3.6}$$

$$\mu_{min} := \min_{u\in[U],k\in[K]} \mu_{uk}; \tag{3.7}$$

$$\mu_{max} := \max_{u\in[U],k\in[K]} \mu_{uk}; \tag{3.8}$$

$$\lambda_{min} := \min_{u\in[U]} \lambda_u. \tag{3.9}$$

The following assumptions will be in force throughout this chapter.

Assumption 3.1 (Optimal Matching)**.** *There is a* unique optimal matching*, i.e.:*

1. *There is a unique optimal server for each queue: k_u^* is a singleton, i.e., $\Delta_{uk} > 0$ for $k \neq k_u^*$, for all u,*

2. *The optimal queue-server pairs for a matching: For any $u' \neq u$, $k_u^* \neq k_{u'}^*$.*

Assumption 3.2 (Stability)**.** *The optimal matching stabilizes every queue, i.e., the arrival rates lie within the stability region: $\epsilon_u > 0$ for all $u \in [U]$.*

The assumption of a unique optimal matching essentially means that the queues and servers are solving a pure coordination problem; for example, in the crowdsourcing example described in the introduction, this would correspond to the presence of a unique worker best suited to each type of job.

We evaluate the performance of scheduling policies against the policy that schedules the optimal matching in every time slot. Let $\mathbf{Q}(t)$ be the queue-length vector at time t under our specified algorithm, and let $\mathbf{Q}^*(t)$ be the corresponding vector under the optimal policy. We define *regret* as the difference in mean queue-lengths for the two policies. That is, the regret (vector) is given by: $\mathbf{\Psi}(t) := \mathbb{E}\left[\mathbf{Q}(t) - \mathbf{Q}^*(t)\right]$. We use the terms *queue-regret* or simply *regret* to refer to $\mathbf{\Psi}(t)$.

Throughout, when we evaluate queue-regret, we do so under the assumption that the queueing system starts in the steady state distribution of the system induced by the optimal policy, as follows.

Assumption 3.3 (Initial State)**.** *The queueing system starts with an initial state $\mathbf{Q}(0)$ distributed according to the stationary distribution of $\mathbf{Q}^*(t)$, i.e., under the policy that chooses the optimal matching at each time step; this distribution is denoted $\pi_{(\boldsymbol{\lambda},\boldsymbol{\mu}^*)}$.*

Table 3.1: General Notation

Symbol	**Description**
λ_u	Expected rate of arrival to queue u
λ_{min}	Minimum arrival rate across all queues
$A_u(t)$	Arrival at time t to queue u
μ_{uk}	Expected service rate of server k for queue u
$R_{uk}(t)$	Service rate between server k queue u at time t
k_u^*	Best server for queue u
μ_u^*	Expected rate of best server for queue u
μ_{max}	Maximum service rate across all links
μ_{min}	Minimum service rate across all links
Δ	Minimum (among all queues) difference between the best and second best servers
$\kappa_u(t)$	server assigned to queue u at time t
$S_u(t)$	Potential service provided by server assigned to queue u at time t
$Q_u(t)$	queue-length of queue u at time t
$Q_u^*(t)$	queue-length of queue u at time t for the optimal strategy
$\Psi_u(t)$	Regret for queue u at time t

3.4 The Late Stage

We analyze the performance of a scheduling algorithm with respect to queue-regret as a function of time and system parameters like

(a) the load on the system $\epsilon := \mu^* - \lambda$, and

(b) the minimum difference between the rates of the best and the next best servers $\Delta := \mu^* - \max_{k \neq k^*} \mu_k$.

As a preview of the theoretical results, Figure 3.1 shows the evolution of queue-regret with time in a system with 5 servers under a scheduling policy inspired by UCB. Exact details of the scheduling algorithm can be found in Section 3.4.2. It is observed that the regret goes through a phase transition. In the initial stage, when the algorithm has not estimated the service rates well enough to stabilize the queue, the regret grows poly-logarithmically similar to the classical MAB setting. After a critical point when the algorithm has learned the system parameters well enough to stabilize the queue, the queue-length goes through regenerative cycles as the queue become empty. Thus at the beginning of every regenerative cycle,

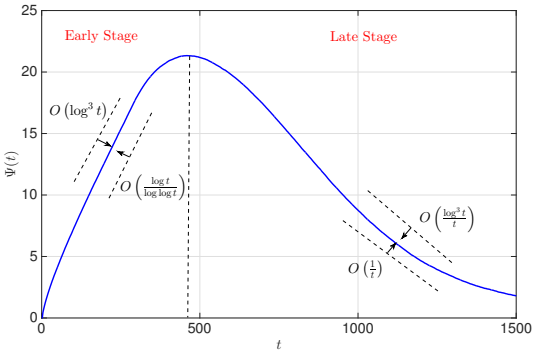


Figure 3.1: Variation of queue-regret $\Psi(t)$ for a particular user under Q-UCB in a 1×5 system with $\epsilon = 0.15$ and $\Delta = 0.17$

there is no accumulation of past errors and the sample-path queue-regret is at most zero. As the algorithm estimates the parameters better with time, the length of the regenerative cycles decreases and the queue-regret decays to zero.

An interesting question to ask is: how does the queue-regret scale as $t \to \infty$? For the classical MAB problem, it is well-known that regret, which is the cumulative sum of the rate loss in each time-slot, scales as $O(\log t)$. However for the queueing bandit, our intuition suggests that the accumulation of errors is only over regenerative cycles. Now observe that the derivative of cumulative regret (which roughly corresponds to regret per time-slot) is $O(1/t)$, and that the regenerative cycle-lengths for the optimal policy are $\Theta(1)$. Thus, it is reasonable to believe that the queue-regret at time t is $O\left(1/t\right)$ times a constant factor that increases with the length of the regenerative cycle.

To push this intuition through to a formal proof requires high probability results for bandits over finite intervals of time corresponding to queue busy periods, where the intervals are random variables which are themselves coupled to the bandit strategy (the regenerative cycle-lengths are coupled to the bandit scheduling decisions). Traditional algorithms like UCB1 and Thompson Sampling guarantee high probability results only after sufficient number of sub-optimal arm pulls [21]. [20] gives upper tail bounds for the number of sub-optimal arm pulls for the UCB1 and UCB-V algorithms but these bounds are polynomial in the factor of deviation from the mean and not polynomial in time. To the best of our knowledge, there is a lack of high probability upper and lower bounds in the multi-armed bandit literature for arbitrary time intervals. The lack of such results motivates us to use alternate proof

strategies for the lower bound on queue-regret and the achievability results. For the lower bound, in Section 3.4.1 we use coupling arguments to derive lower bounds on queue regret growth over one time-step for any α-consistent policy. This allows us to show that no α-consistent policy can achieve a better scaling than $O\left(1/t\right)$. For the achievability result, in Section 3.4.2 we construct a structured exploration variant of Thompson Sampling (a combination of ϵ-greedy and Thompson Sampling algorithms) that results in upper bounds on the expected number of sub-optimal schedules over finite time intervals. When combined with queueing arguments, this leads to an upper bound on queue regret.

Remark 3.1. *All the results derived in this work for Q-UCB also hold for a similar structured-explore variant of Thompson Sampling, which we refer to as Q-ThS. The Q-ThS algorithm is presented in detail in Appendix B.*

Notation: For the results in Section 3.4, the notation $f(t) = O\left(g(K, \epsilon, t)\right)$ for all $t \in h(K, \epsilon)$ (here, $h(K, \epsilon)$ is an interval that depends on K, ϵ) implies that there exist constants C and t_0 independent of K and ϵ such that $f(t) \leq Cg(K, \epsilon, t)$ for all $t \in (t_0, \infty) \cap h(K, \epsilon)$.

3.4.1 An Asymptotic Lower Bound

We establish an asymptotic lower bound on regret for the class of α-consistent policies; this class for the queueing bandit is a generalization of the α-consistent class used in the literature for the traditional stochastic MAB problem [91, 120, 41]. The precise definition is given below.

Definition 3.1. *A scheduling policy is said to be α-consistent (for some $\alpha \in (0,1)$) if given a problem instance $(\lambda, \boldsymbol{\mu})$,*

$$\mathbb{E}\left[\sum_{s=1}^{t} \mathbb{1}\{\kappa(s) = k\}\right] = O(t^{\alpha})$$

for all $k \neq k^*$.

This means that any policy under this class schedules a sub-optimal server not more than $O(t^{\alpha})$ (sub-linear in t) number of times. Without a restriction such as that imposed in the preceding definition, "trivial" policies that, for e.g., schedule the same server every time step would be allowed. Such policies would have zero expected regret if the chosen server happened to be optimal, and otherwise would have linear regret. The α-consistency requirement rules out such policies, while ensuring the set under consideration is reasonable.

Theorem 3.1 below gives an asymptotic lower bound on the average queue-regret and per-queue regret for an arbitrary α-consistent policy.

Theorem 3.1. *For any problem instance $(\lambda, \boldsymbol{\mu})$ and any α-consistent policy, the regret $\Psi(t)$ satisfies*

$$\Psi(t) \geq \left(\frac{\lambda}{4} D(\boldsymbol{\mu})(1-\alpha)(K-1)\right)\frac{1}{t}$$

for infinitely many t, where

$$D(\boldsymbol{\mu}) = \frac{\Delta}{\mathrm{KL}\left(\mu_{min}, \frac{\mu^*+1}{2}\right)}. \tag{3.10}$$

Outline for theorem 3.1. The proof of the lower bound consists of three main steps. First, in lemma B.11, we show that the regret at any time-slot is lower bounded by

the probability of a sub-optimal schedule in that time-slot (up to a constant factor that is dependent on the problem instance). The key idea in this lemma is to show the equivalence of any two systems with the same marginal service distributions with respect to bandit algorithms. This is achieved through a carefully constructed coupling argument that maps the original system with independent service across links to another system with service process that is dependent across links but with the same marginal distribution.

As a second step, the lower bound on the regret in terms of the probability of a sub-optimal schedule enables us to obtain a lower bound on the cumulative queue-regret in terms of the number of sub-optimal schedules. We then use a lower bound on the number of sub-optimal schedules for α-consistent policies (lemma B.10 and corollary B.1) to obtain a lower bound on the cumulative regret. In the final step, we use the lower bound on the cumulative queue-regret to obtain an *infinitely often* lower bound on the queue-regret. □

3.4.2 Achieving the Asymptotic Bound

Theorem 3.1 implies that no α-consistent policy can achieve a queue-regret better than $O(1/t)$. We next ask if straight-forward generalizations of standard bandit algorithms like UCB and Thompson sampling can achieve a scaling of $O\left(1/t\right)$, thus matching the lower bound in theorem 3.1. To prove that these algorithms achieve this scaling, it is essential to show high probability bounds on scheduling errors over regenerative cycles in the late stage. A systematic way to show this would be to prove that the algorithm has a good estimate of all the link rates in

the late stage leading to the correct scheduling decision. But for standard bandit algorithms, lack of concentration results on the number of times each link is scheduled makes it difficult to prove a high probability bound on scheduling errors over a finite time-interval in the late stage.

To get around this difficulty, we propose a slightly modified version of the UCB1 algorithm generalized to the multi-dimensional queueing bandit. The algorithm, which we call Q-UCB, has an explicit structured exploration component similar to ϵ-greedy algorithms. The structured exploration ensures that the algorithm has a sufficiently good estimate of all the link rates (including sub-optimal ones) in the late stage.

We now describe the algorithm we employ in detail. Let $\mathcal{M} \subset [K]^U$ be the set of all matchings[4]. We represent a matching by a U-length vector in which the u^{th} element gives the server that is assigned to queue u. A vector $\boldsymbol{\kappa} \in \mathcal{M}$ if and only if for any $u' \neq u$, $\kappa_{u'} \neq \kappa_u$. Let $\mathcal{E} \subset \mathcal{M}$ be a subset of K perfect matchings such that their union covers the set of all edges in the complete bipartite graph[5]. Also, let $T_{uk}(t)$ be the number of times server k is assigned to queue u in the first t time-slots and $\hat{\boldsymbol{\mu}}(t)$ be the empirical mean of service rates at time-slot t from past observations (until $t-1$).

At time-slot t, Q-UCB decides to *explore* with probability $\min\{1, 3K\log^2 t/t\}$, otherwise it *exploits*. To explore, it chooses a matching uniformly at random from

[4]A matching is given by a set of edges such that every node in the graph is incident to at most one edge.

[5]It is easy to show that such a decomposition is possible.

Table 3.2: Notation specific to Algorithm 2

Symbol	Description
$\mathsf{E}(t)$	Indicates if the algorithm schedules a matching through *Explore*
$\mathsf{E}_{uk}(t)$	Indicates if Server k is assigned to Queue u at time t through *Explore*
$\mathsf{I}_{uk}(t)$	Indicates if Server k is assigned to Queue u at time t through *Exploit*
$T_{uk}(t)$	Number of time slots Server k is assigned to Queue u in time $[1, t-1]$
$\hat{\boldsymbol{\mu}}(t)$	Empirical mean of service rates at time t from past observations (until $t-1$)
$\boldsymbol{\kappa}(t)$	Matching scheduled in time-slot t

the set $\mathcal{E}$. The chosen exploration rate ensures that we are able to obtain concentration results for the number of times any link is sampled.[6] If it exploits, it makes a scheduling decision based on upper confidence bounds for the link rates. Specifically, it first determines for every queue, the server that has the highest upper confidence bound for the corresponding link rate. It then schedules the projection of this U-length server vector onto the space of all matchings $\mathcal{M}$. Notation and details of the algorithm are given in Table 3.2 and Algorithm 2 respectively.

We now show that, for a given problem instance $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ (and therefore fixed $\boldsymbol{\epsilon}$), the regret under Q-UCB scales as $O\left(\text{poly}(\log t)/t\right)$. We state the most general form of the asymptotic upper bound in theorem 3.2. A slightly weaker version of the result is given in corollary 3.1. This corollary is useful to understand the dependence of the upper bound on the load $\boldsymbol{\epsilon}$ and the number of servers K.

[6]The exploration rate could scale like $\log t/t$ if we knew Δ in advance; however, without this knowledge, additional exploration is needed.

Algorithm 2 Q-UCB

At time t,

Let $\mathsf{E}(t)$ be an independent Bernoulli sample of mean $\min\{1, 3K\frac{\log^2 t}{t}\}$.

if $\mathsf{E}(t) = 1$ **then**

Explore:

Schedule a matching from $\mathcal{E}$ uniformly at random.

else

Exploit:

Compute for all $u \in [U]$

$$\hat{k}_u(t) := \arg\max_{k\in[K]} \hat{\mu}_{uk}(t) + \sqrt{\frac{\log^2 t}{2T_{uk}(t-1)}}.$$

Schedule a matching $\boldsymbol{\kappa}(t)$ such that

$$\boldsymbol{\kappa}(t) \in \arg\min_{\boldsymbol{\kappa}\in\mathcal{M}} \sum_{u\in[U]} \mathbb{1}\left\{\kappa_u \neq \hat{k}_u(t)\right\},$$

i.e., $\boldsymbol{\kappa}(t)$ *is the projection of* $\hat{\boldsymbol{k}}(t)$ *onto the space of all matchings* $\mathcal{M}$ *with Hamming distance as metric.*

end if

Remark 3.2. *For any queue u, we state the regret bounds and the corresponding time-intervals in which these bounds hold as a function of ϵ_u, the gap between the arrival rate and the best service rate for that queue. Therefore, the time ranges for which the bounds hold may vary for different queues depending on $\boldsymbol{\epsilon}$.*

Theorem 3.2. *Consider any problem instance $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ which has a single best matching. For any $u \in [U]$, let $w(t) = \exp\left(\left(\frac{2\log t}{\Delta}\right)^{2/3}\right)$, $v'_u(t) = \frac{6K}{\epsilon_u} w(t)$ and $v_u(t) = \frac{24}{\epsilon_u^2}\log t + \frac{60K}{\epsilon_u}\frac{v'_u(t)\log^2 t}{t}$. Then, under Q-UCB the regret for queue u, $\Psi_u(t)$, satisfies*

$$\Psi_u(t) = O\left(\frac{K v_u(t)\log^2 t}{t}\right)$$

for all t such that $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$ and $v_u(t) + v'_u(t) \leq t/2$.

Corollary 3.1. *Let $w(t) = \exp\left(\left(\frac{2\log t}{\Delta}\right)^{2/3}\right)$. Then,*

$$\Psi_u(t) = O\left(K\frac{\log^3 t}{\epsilon_u^2 t}\right)$$

for all t such that $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$, $\frac{t}{w(t)} \geq \max\left\{\frac{24K}{\epsilon_u}, 15K^2\log t\right\}$, and $\frac{t}{\log t} \geq \frac{198}{\epsilon_u^2}$.

Outline for Theorem 3.2. As mentioned earlier, the central idea in the proof is that the sample-path queue-regret is at most zero at the beginning of regenerative cycles, i.e., instants at which the queue becomes empty. The proof consists of two main parts – one which gives a high probability result on the number of sub-optimal schedules in the exploit phase in the late stage, and the other which shows that at any time, the beginning of the current regenerative cycle is not very far in time.

The former part is proved in lemma B.2, where we make use of the structured exploration component of Q-UCB to show that all the links, including the sub-optimal ones, are sampled a sufficiently large number of times to give a good estimate

of the link rates. This in turn ensures that the algorithm schedules the correct links in the exploit phase in the late stages with high probability.

For the latter part, we prove a high probability bound on the last time instant when the queue was zero (which is the beginning of the current regenerative cycle) in lemma B.8. Here, we make use of a recursive argument to obtain a tight bound. More specifically, we first use a coarse high probability upper bound on the queue-length (lemma B.4) to get a first cut bound on the beginning of the regenerative cycle (lemma B.5). This bound on the regenerative cycle-length is then recursively used to obtain tighter bounds on the queue-length, and in turn, the start of the current regenerative cycle (lemmas B.7 and B.8 respectively).

The proof of the theorem proceeds by combining the two parts above to show that the main contribution to the queue-regret comes from the structured exploration component in the current regenerative cycle, which gives the stated result. □

This result, in combination with theorem 3.1, shows that queue-regret for Q-UCB in the long-term is within a poly($\log t$) factor of the optimal queue-regret for the α-consistent class.

Remark 3.3. *Although we assume Bernoulli distributions for arrival and service rates in our model, the result in theorem 3.2 holds for general, non-Bernoulli distributions with bounded support if*

(i) there is a unique matching that gives the best mean-rate for all users (similar to assumption 3.1 in Section 3.3), and

(ii) the genie policy that defines the regret $\mathbf{\Psi}(t)$ is the one that always schedules the best mean-rate matching.

3.5 The Early Stage in the Heavily Loaded Regime

Theorem 3.2 shows that systematic exploration in Q-UCB ensures an $O\left(\text{poly}(\log t)/t\right)$ queue-regret in the late stage. The penalty for aggressive exploration is likely to be more apparent in the initial stages when the queues have not yet stabilized and there are few regenerative cycles. As a result, the queueing system has a behavior similar to the traditional MAB system in the early stage. Thus, it is reasonable to expect that algorithms that achieve optimal performance for the traditional MAB problem also perform well in the early stages in the queueing system.

In order to study the performance of α-consistent policies in the early stage, we consider the *heavily loaded* system, where the arrival rate λ is close to the optimal service rate μ^*. Specifically, we characterize the behavior of queue-regret as the difference between the two rates, $\epsilon = \mu^* - \lambda \to 0$.

Analyzing regret in the early stage in the heavily loaded regime has the effect that the the optimal server is the only one that stabilizes the queue. As a result, in the heavily loaded regime, effective learning and scheduling of the optimal server play a crucial role in determining the transition point from the early stage to the late stage. For this reason the heavily loaded regime reveals the behavior of regret in the early stage.

Notation: For all the results in this section, the notation $f(t) = O\left(g(K, \epsilon, t)\right)$ for all $t \in h(K, \epsilon)$ ($h(K, \epsilon)$ is an interval that depends on K, ϵ) implies that there exist numbers C and ϵ_0 that depend on Δ such that for all $\epsilon \geq \epsilon_0$, $f(t) \leq Cg(K, \epsilon, t)$ for all $t \in h(K, \epsilon)$.

Theorem 3.3 gives a lower bound on the regret in the heavily loaded regime, roughly in the time interval $\left(K^{1/1-\alpha}, O\left(K/\epsilon\right)\right)$ for any α-consistent policy.

Theorem 3.3. *Given any problem instance $(\lambda, \boldsymbol{\mu})$, and for any α-consistent policy and $\gamma > \frac{1}{1-\alpha}$, the regret $\Psi(t)$ satisfies*

$$\Psi(t) \geq \frac{D(\boldsymbol{\mu})}{2}(K-1)\frac{\log t}{\log\log t}$$

for $t \in \left[\max\{C_1 K^{\gamma}, \tau\}, (K-1)\frac{D(\boldsymbol{\mu})}{2\epsilon}\right]$ where $D(\boldsymbol{\mu})$ is given by equation 3.10, and τ and C_1 are constants that depend on α, γ and the policy.

Outline for Theorem 3.3. The crucial idea in the proof is to show a lower bound on the queue-regret in terms of the number of sub-optimal schedules (Lemma B.12). As in Theorem 3.1, we then use a lower bound on the number of sub-optimal schedules for α-consistent policies (given by Corollary B.1) to obtain a lower bound on the queue-regret. □

Theorem 3.3 shows that, for any α-consistent policy, it takes at least $\Omega\left(K/\epsilon\right)$ time for the queue-regret to transition from the early stage to the late stage. In this region, regret is growing with time, and the scaling $O(\log t/\log\log t)$ reflects the fact that in this regime queue-regret is dominated by the fact that cumulative

regret grows like $O(\log t)$. A reasonable question then arises: after time $\Omega\left(K/\epsilon\right)$, should we expect the regret to transition into the late stage regime analyzed in the preceding section?

We answer this question by studying when Q-UCB achieves its late-stage regret scaling of $O\left(\text{poly}(\log t)/\epsilon^2 t\right)$ scaling; as we will see, in an appropriate sense, Q-UCB is close to optimal in its transition from early stage to late stage, when compared to the bound discovered in Theorem 3.3. Formally, we have Corollary 3.2, which is an analog to Corollary 3.1 under the heavily loaded regime.

Corollary 3.2. *For any problem instance* $(\lambda, \boldsymbol{\mu})$*, any* $\gamma \in (0,1)$ *and* $\delta \in (0, \min(\gamma, 1-\gamma))$*, the regret under Q-UCB satisfies*

$$\Psi(t) = O\left(\frac{K \log^3 t}{\epsilon^2 t}\right)$$

$\forall t \geq C_2 \max\left\{\left(\frac{1}{\epsilon}\right)^{\frac{1}{\gamma-\delta}}, \left(\frac{K}{\epsilon}\right)^{\frac{1}{1-\gamma}}, (K^2)^{\frac{1}{1-\gamma-\delta}}, \left(\frac{1}{\epsilon^2}\right)^{\frac{1}{1-\delta}}\right\}$*, where* C_2 *is a constant independent of* ϵ *(but depends on* Δ*,* γ *and* δ*).*

By combining the result in Corollary 3.2 with Theorem 3.3, we can infer that in the heavily loaded regime, the time taken by Q-UCB to achieve $O\left(\text{poly}(\log t)/\epsilon^2 t\right)$ scaling is, in some sense, order-wise close to the optimal in the α-consistent class. Specifically, for any $\beta \in (0,1)$, there exists a scaling of K with ϵ such that the queue-regret under Q-UCB scales as $O\left(\text{poly}(\log t)/\epsilon^2 t\right)$ for all $t > (K/\epsilon)^\beta$ while the regret under any α-consistent policy scales as $\Omega\left(K \log t/\log\log t\right)$ for $t < K/\epsilon$.

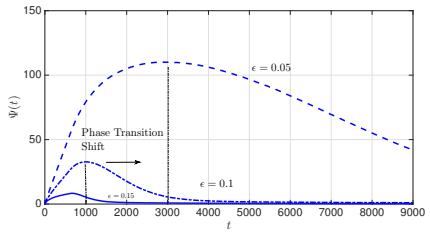
We conclude by noting that while the transition point from the early stage to the late stage for Q-UCB is near optimal in the heavily loaded regime, it does does

not yield optimal regret performance in the early stage in general. In particular, recall that at any time t, the structured exploration component in Q-UCB is invoked with probability $3K \log^2 t/t$. As a result, we see that, in the early stage, queue-regret under Q-UCB could be a $\log^2 t$-factor worse than the $\Omega(\log t/\log\log t)$ lower bound shown in Theorem 3.3 for the α-consistent class. This intuition can be formalized: it is straightforward to show an upper bound of $2K \log^3 t$ for any $t > \max\{C_3, U\}$, where C_3 is a constant that depends on Δ but is independent of K and ϵ; we omit the details.

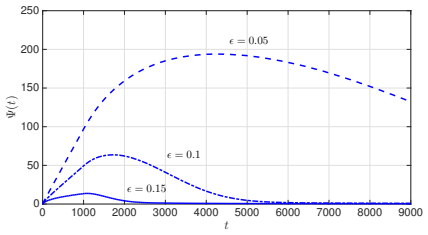
3.6 Simulation Results

In this section we present simulation results of various queueing bandit systems with K servers. These results corroborate our theoretical analysis in Sections 3.4 and 3.5. In particular a phase transition from unstable to stable behavior can be observed in all our simulations, as predicted by our analysis. In the remainder of the section we demonstrate the performance of Algorithm 4 under variations of system parameters like the traffic (ϵ), the gap between the optimal and the suboptimal servers (Δ), and the size of the system (K). We also compare the performance of our algorithm with versions of UCB-1 [21] and Thompson Sampling [132] without structured exploration (Figure 3.3 in the appendix).

Variation with ϵ and K. In Figure 3.2 we see the evolution of $\Psi(t)$ in systems of size 5 and 7 . It can be observed that the regret decays faster in the smaller system, which is predicted by Theorem 3.2 in the late stage and Corollary 3.2 in the early stage. The performance of the system under different traffic settings can be



(a) Queue-Regret under Q-ThS for a system with 5 servers with $\epsilon \in \{0.05, 0.1, 0.15\}$



(b) Queue-Regret under Q-ThS for a a system with 7 servers with $\epsilon \in \{0.05, 0.1, 0.15\}$

Figure 3.2: Variation of Queue-regret $\Psi(t)$ with K and ϵ under Q-Ths. The phase-transition point shifts towards the right as ϵ decreases. The efficiency of learning decreases with increase in the size of the system.

observed in Figure 3.2. It is evident that the regret of the queueing system grows with decreasing ϵ. This is in agreement with our analytical results (Corollaries 3.1 and 3.2). In Figure 3.2 we can observe that the time at which the phase transition occurs shifts towards the right with decreasing ϵ which is predicted by Corollaries 3.1 and 3.2.

3.7 Summary and Discussion

This work provides the first regret analysis of the queueing bandit problem; as emphasized, it is quite different than that seen for minimization of cumulative regret in the standard MAB problem. Our main results provide a comprehensive analysis of the behavior of regret in two regimes: the late stage, once the bandit queueing system has regenerative $\Theta(1)$ cycles; and the early stage, before the bandit queueing system has been stabilized. Our main findings are as follows. *First*, we show that asymptotically, queue-regret is at least $O(1/t)$ infinitely often under

any α-consistent policy. Further, we show that we can achieve this lower bound (to poly-logarithmic factors), using a variant of UCB with the addition of structured exploration (Q-UCB). *Second*, by analyzing the queueing regret in the heavily loaded regime, we show that in the early stage, queue-regret is lower bounded by $O(\log t/\log\log t)$, because the queue still exhibits long regenerative cycles in this regime. *Third*, we show that the upper bound on regret of Q-UCB "switches" to the late stage from the early stage at a nearly optimal time.

Our analysis highlights why minimization of queue-regret presents a subtle learning problem. On one hand, if the queue has been stabilized, the presence of regenerative cycles allows us to establish that queue regret must eventually decay to zero at rate $1/t$ under an optimal algorithm (the late stage). On the other hand, to actually have regenerative cycles in the first place, a learning algorithm needs to exploit enough to actually stabilize the queue (the early stage). Our analysis not only characterizes regret in both regimes, but also characterizes the transition point between the two regimes. In this way the queueing bandit is a remarkable new example of the tradeoff between exploration and exploitation.

Is structured exploration necessary?

We conclude by discussing a key feature of our technical analysis. Crucially, to prove our results, we need concentration results on regret over *finite interval of time,* corresponding to the stochastic renewal cycles of queues. These stochastic time intervals over which we need concentrations are endogenously related to the wrong arm pull probabilities, because both are determined by the bandit strategy.

The main impact of our use of structured exploration in Q-UCB is to allow us to attain the optimal asymptotic regret for the queueing bandit, by enabling high probability[7] results. However, as noted above, in the early stage the cost of our use of structured exploration is a cubic-logarithmic factor. Standard UCB or Thompson sampling, on the other hand, provide a logarithmic cumulative regret, and so ensure that queue-regret is logarithmic in the early stage; however, due to the lack of concentration results, especially over the stochastically coupled finite time intervals for these methods, we cannot show the more critical $1/t$ regret scaling in the late stage for either algorithm.

Is Thompson Sampling optimal?

The above discussion prompts an interesting open direction for future work: *Can bandit algorithms that are optimal for the standard MAB problem (e.g., UCB and Thompson sampling) achieve a queue-regret scaling as* $\text{poly}(\log t)/t$ *for all sufficiently large* t*?* We explore this question empirically with the simulation study below.

Figure 3.3 shows a comparison of the performance of structured explore variants of UCB and Thompson Sampling (Q-UCB, Q-ThS) against their traditional (unstructured) counterparts. The figure contains two plots of Q-ThS, one with exploration probability $3K\log^2 t/t$ (as suggested by the theoretical analysis) and another with exploration probability of $0.4K\log^2 t/t$. It can be observed that in the early stage, the unstructured algorithms perform better which is an artifact of

[7] bad event probability decaying at least polynomially with time

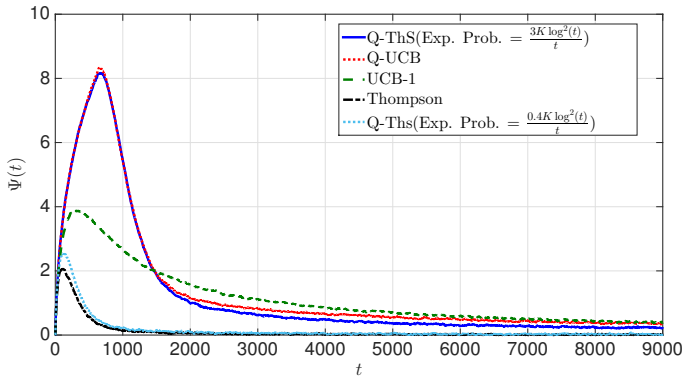


Figure 3.3: Comparison of queue-regret performance of Q-ThS, Q-UCB, UCB-1 and Thompson Sampling in a 5 server system with $\epsilon_u = 0.15$ and $\Delta = 0.17$. Two variants of Q-ThS are presented, with different exploration probabilities; note that $3K\log^2 t/t$ is the exploration probability suggested by theoretical analysis (which is necessarily conservative). Tuning the constant significantly improves performance of Q-ThS.

the extra exploration required by Q-UCB and Q-ThS. In the late stage we observe that Q-UCB gives marginally better performance than UCB-1, however traditional Thompson sampling has the best performance in both stages. Q-ThS is dominated as well, but can be made to nearly match Thompson sampling by tuning the exploration probability.

Nevertheless, it appears that Thompson sampling dominates UCB-1, Q-UCB, and the theoretically analyzed version of Q-ThS, at least over the finite time intervals considered. Similar empirical observations in standard bandit problems [39, 122] have led to a surge of interest in Thompson sampling. Given these numerical experiments, it is important to understand whether theoretical regret bounds can be established for Thompson sampling (e.g., in the spirit of the analysis in [76, 13, 119]).

We believe that sharp concentration results (with polynomially decaying probability for a sub-optimal arm pull in a finite time period) are crucial for proving upper bounds on queue-regret for any algorithm. A related issue also shows up in our late stage lower bound: if we had per-time-slot lower bound (which is a type of finite time concentration) on the sub-optimal arm pull probabilities for the class of all α-consistent bandit policies, it would be possible to show a C/t decay in queue-regret for all t large enough. As bandit algorithms do not (in general) have a high probability per-time-slot lower bound on wrong arm pull probabilities, we are instead able to show a bound of C/t infinitely often (as opposed to "for all sufficiently large times"). In general, it is not clear if finite time concentration results (either upper or lower bounds) are possible.

Chapter 4

Scheduling with Energy Costs

4.1 Introduction

Due to the tremendous increase in demand for data traffic, modern cellular networks have taken the densification route to support peak traffic demand [28]. While increasing the density of base-stations gives greater spectral efficiency, it also results in increased costs of operating and maintaining the deployed base-stations. Rising energy cost is a cause for concern, not only from an environmental perspective, but also from an economic perspective for network operators as it constitutes a significant portion of the operational expenditure[1]. To address this challenge, latest research aims to design energy efficient networks that balance the trade-off between spectral efficiency, energy efficiency and user QoS requirements [141, 111].

Studies reveal that base-stations contribute to more than half of the energy consumption in cellular networks [103, 142]. Although dense deployment of base-stations are useful in meeting demand in peak traffic hours, they regularly have excess capacity during off-peak hours [111, 74]. A fruitful way to conserve power is, therefore, to dynamically switch off under-utilized base-stations. For this purpose,

[1]Subhashini Krishnasamy, Akhil Padinhare Thalasseryveettil, Ari Arapostathis, Sanjay Shakkottai, and Rajesh Sundaresan. "Augmenting MaxWeight with Explicit Learning for Wireless Scheduling with Switching Costs". In the Proceedings of the 36th Annual IEEE International Conference on Computer Communications, INFOCOM 2017. Co-authors of the paper made equal contributions in obtaining these results.

modern cellular standards incorporate protocols that include *sleep* and *active* modes for base-stations. The sleep mode allows for selectively switching under-utilized base-stations to low energy consumption modes. This includes completely switching off base-stations or switching off only certain components.

Consider a time-slotted multi base-station (BS) cellular network where subsets of BSs can be dynamically activated. Since turning off BSs could adversely impact the performance perceived by users, it is important to consider the underlying energy vs. performance trade-off in designing BS activation policies. In this chapter, we study the joint problem of dynamically selecting the BS activation sets and user rate allocation depending on the network load. We take into account two types of overheads involved in implementing different activation modes in the BSs.

(i) Activation cost occurs due to maintaining a BS in the active state. This includes energy spent on main power supply, air conditioning, transceivers and signal processing [74]. Surveys show that a dominant part of the energy consumption of an active base-station is due to static factors that do not have dependencies with traffic load intensities [111, 17]. Therefore, an active BS consumes almost the same energy irrespective of the amount of traffic it serves. Typically, the operation cost (including energy consumption) in the sleep state is much lower than that in the active state since it requires only minimal maintenance signaling [142].

(ii) Switching cost is the penalty due to switching a BS from active state to sleep state or vice-versa. This factors in the signaling overhead (control signaling to users, signaling over the backhaul to other BSs and/or the BS controller), state-migration processing, and switching energy consumption associated with dynamically changing

the BS modes [74].

Further, switching between these states typically cannot occur instantaneously. Due to the hysteresis time involved in migrating between the active and sleep states, BS switching can be done only at a slower time-scale than that of channel scheduling [8, 147].

Main Contributions

We formulate the problem in a (stochastic) network cost minimization framework. The task is to select the set of active BSs in every time-slot, and then based on the instantaneous channel state for the activated BSs, choose a feasible allocation of rates to users. Our aim is to minimize the total network cost (sum of activation and switching costs) subject to stability of the user queues at the BSs.

Insufficiency of the standard Lyapunov technique: Such stochastic network resource allocation problems typically adopt greedy primal dual algorithms along with virtual-queues to accommodate resource constraints [56, 96, 127]. To ensure stability, this technique crucially relies on achieving negative Lyapunov drift in every time-slot (or within some finite number of time-slots). In our problem, unlike in the traditional setting, such an approach cannot be applied for two reasons. *First*, introduction of switching cost makes it a non-convex problem due to ergodicity constraints. *Second*, switching cost introduces challenges in showing Lyapunov stability for one-step greedy Lyapunov based algorithms. Lyapunov stability is shown in the traditional setting because the channel state in every time-slot is independent of the controller's actions, and therefore, provides an average (potential) service rate

that is strictly higher than the average arrival rate. But in the current problem, effective channel state in each time-slot (consisting of feasible rates for the activated BS set) is determined by the BS activation decision in that time-slot. Since switching cost depends on the change in activation state in consecutive time-slots, traditional virtual-queue based algorithms introduce coupling of activation decisions across time. Thus, the evolution of the effective channel rates are dependent across time through the scheduling decisions, and this results in co-evolution of packet queues and the channel state distribution.

To circumvent difficulties introduced through this co-evolution, we propose an approach that uses queue-lengths for channel scheduling at a fast time-scale, but explicitly uses arrival and channel statistics (using learning via an explore-exploit learning policy) for activation set scheduling at a slower time-scale. Our main contributions are as follows.

1. **Static-split Activation + Max-Weight Channel Scheduling:** We propose a solution that explicitly controls the time-scale separation between BS activation and rate allocation decisions. At BS switching instants (which occurs at a slow time-scale), the strategy uses a static-split rule (time-sharing) which is pre-computed using the explicit knowledge of the arrival and channel statistics for selecting the activation state. This activation algorithm is combined with a queue-length based Max-Weight algorithm for rate allocation (applied at the fast time-scale of channel variability). We show that the joint dynamics of these two algorithms leads to stability; further, the choice

of parameters for the activation algorithm enables us to achieve an average network cost that can be made arbitrarily close to the optimal cost.

2. **Learning algorithm with provable guarantees:** In the setting where the arrival and channel statistics are not known, we propose an *explore-exploit* policy that estimates arrival and channel statistics in the explore phase, and uses the estimated statistics for activation decisions in the exploit phase (this phase includes BS switching at a slow time-scale). This is combined with a Max-Weight based rate allocation rule restricted to the activated BSs (at a fast time-scale). We prove that this joint learning-cum-scheduling algorithm can ensure queue stability while achieving close to optimal network cost.

3. **Convergence bounds for time-inhomogeneous Markov chains:** In the course of proving the theoretical guarantees for our algorithm, we derive useful technical results on convergence of time-inhomogeneous Markov chains. More specifically, we derive explicit convergence bounds for the marginal distribution of a finite-state time-inhomogeneous Markov chain whose transition probability matrices at each time-step are arbitrary (but small) perturbations of a given stochastic matrix. We believe that these bounds are useful not only in this specific problem, but are of independent interest.

To summarize then, our approach can be viewed as an algorithmically engineered separation of time-scales for only the activation set dynamics, while adapting to the channel variability for the queue dynamics. Such an engineering of time-scales leads to coupled fast-slow dynamics, the 'fast' due to opportunistic channel

allocation and packet queue evolution with Max-Weight, and the 'slow' due to infrequent base-station switching using learned statistics. Through a novel Lyapunov technique for convergent time-inhomogeneous Markov chains, we show that we can achieve queue stability while operating at a near-optimal network cost.

Related Work

While mobile networks have been traditionally designed with the objective of optimizing spectral efficiency, design of energy efficient networks has been of recent interest. A survey of various techniques proposed to reduce operational costs and carbon footprint can be found in [111, 66, 141, 142]. The survey in [142] specially focuses on sleep mode techniques in BSs.

Various techniques have been proposed to exploit BS sleep mode to reduce energy consumption in different settings. Most of them aim to minimize energy consumption while guaranteeing minimum user QoS requirements. For example, [65, 74, 60] consider inelastic traffic and consider outage probability or blocking probability as metrics for measuring QoS. In [8], the problem is formulated as a utility optimization problem with the constraint that the minimum rate demand should be satisfied. But they do not explicitly evaluate the performance of their algorithm with respect to user QoS. The authors in [75, 63] model a single BS scenario with elastic traffic as an M/G/1 vacation queue and characterize the impact of sleeping on mean user delay and energy consumption. In [147], the authors consider the multi BS setting with Poisson arrivals and delay constraint at each BS.

Most papers that study BS switching use models that ignore switching cost.

Nonetheless, a few papers acknowledge the importance of avoiding frequent switching. For example, Oh et al. [112] implement a hysteresis time for switching in their algorithm although they do not consider it in their theoretical analysis. Gou et al. [63] also study hysteresis sleeping schemes which enforce a minimum sleeping time. In [8] and [147], it is ensured that interval between switching times are large enough to avoid overhead due to transient network states. Finally Jie et al. [74] consider BS sleeping strategies which explicitly incorporate switching cost in the model (but they do not consider packet queue dynamics). They emphasize that frequent switching should be avoided considering its effect on signaling overhead, device lifetime and switching energy consumption, and also note that incorporating switching cost introduces time correlation in the system dynamics.

Notation Important notation for the problem setting can be found in Table 4.1. Boldface letters are used to denote vectors or matrices and the corresponding non-bold letters to denote their individual components. Also, the notation $\mathbb{1}\{\cdot\}$ is used to denote the indicator function. For any two vectors $\mathbf{v}_1$, $\mathbf{v}_2$ and scalar a, $\mathbf{v}_1 \cdot \mathbf{v}_2$ denotes the dot product between the two vectors and $\mathbf{v}_1 + a = \mathbf{v}_1 + a\mathbf{1}$.

4.2 System Model

We consider a time-slotted cellular network with n users and M base-stations (BS) indexed by $u = 1, \ldots, n$ and $m = 1, \ldots, M$ respectively. Users can possibly be connected to multiple BSs. It is assumed that the user-BS association does not vary with time.

4.2.1 Arrival and Channel Model

Data packets destined for a user u arrive at a connected BS m as a bounded (at most $\bar{\mathsf{A}}$ packets in any time-slot), i.i.d. process $\{A_{m,u}(t)\}_{t\geq 1}$ with rate $\mathbb{E}\left[A_{m,u}(t)\right] = \lambda_{m,u}$. Arrivals get queued if they are not immediately transmitted. Let $Q_{m,u}(t)$ represent the queue-length of user u at BS m at the beginning of time-slot t.

The channel between the BSs and their associated users is also time-varying and i.i.d across time (but can be correlated across links), which we represent by the network channel-state process $\{H(t)\}_{t>0}$. At any time t, $H(t)$ can take values from a finite set $\mathcal{H}$ with probability mass function given by $\boldsymbol{\mu}$. Let $\bar{\mathsf{R}}$ be the maximum number of packets that can be transmitted over any link in a single time-slot. We consider an abstract model for interference by defining the set $\mathcal{R}(\mathbf{1}, h)$ as the set of all possible rate vectors when the channel state is h.

4.2.2 Resource Allocation

At any time-slot t, the scheduler has to make two types of allocation decisions:

BS Activation: Each BS can be scheduled to be in one of the two states, *ON* (*active* mode) and *OFF* (*sleep* mode). Packet transmissions can be scheduled only from BSs in the *ON* state. The cost of switching a BS from *ON* in the previous time-slot to *OFF* in the current time-slot is given by C_0 and the cost of maintaining a BS in the *ON* state in the current time-slot is given by C_1. The activation state at time t is denoted by $\mathbf{J}(t) = (J_m(t))_{m\in[M]}$, where $J_m(t) := \mathbb{1}\{\text{BS } m \text{ is } ON \text{ at time } t\}$. We also denote the set of all possible activation states by $\mathcal{J}$. The total cost of

operation, which we refer to as the *network cost*, at time t is the sum of switching and activation cost and is given by

$$C(t) := \mathsf{C}_0 \|(\mathbf{J}(t-1) - \mathbf{J}(t))^+\|_1 + \mathsf{C}_1 \|\mathbf{J}(t)\|_1. \tag{4.1}$$

It is assumed that the current network channel-state $H(t)$ is unavailable to the scheduler at the time of making activation decisions.

Rate Allocation: The network channel-state is observed after the BSs are switched *ON* and before the packets are scheduled for transmission. Moreover, only the part of the channel state restricted to the activated BSs, which we denote by $H(t)|_{\mathbf{J}(t)}$, can be observed. For any $j \in \mathcal{J}, h \in \mathcal{H}$, let $\mathcal{R}(j,h) \subset \mathbb{R}^{M \times n}$ be the set of all possible service rate vectors that can be allocated when the activation set is j and the channel state is h. Given the channel observation $H(t)|_{\mathbf{J}(t)}$, the scheduler allocates a rate vector $\mathbf{S}(t) = (S_{m,u}(t))_{m\in[M],u\in[n]}$ from the set $\mathcal{R}(\mathbf{J}(t), H(t))$ for packet transmission. This allows for draining of $S_{m,u}(t)$ packets from user u's queue at BS m for all $u \in [n]$ and $m \in [M]$.

Thus the resource allocation decision in any time-slot t is given by the tuple $(\mathbf{J}(t), \mathbf{S}(t))$. The sequence of operations in any time-slot can, thus, be summarized as follows: (i) Arrivals, (ii) BS Activation-Deactivation, (iii) Channel Observation, (iv) Rate Allocation, and (v) Packet Transmissions.

4.2.3 Model Extensions

Some of the assumptions in the model above are made for ease of exposition and can be extended in the following ways without affecting the results in this chapter:

Table 4.1: General Notation

Symbol	Description
n	Number of users
M	Number of BSs
$A_{m,u}(t)$	Arrival for user u at BS m at time t
$\bar{\mathsf{A}}$	Maximum number of arrivals to any queue in a time-slot
$\boldsymbol{\lambda}$	Average arrival rate vector
$H(t)$	Channel state at time t
$\mathcal{H}$	Set of all possible channel states
$\boldsymbol{\mu}$	Probability mass function of channel state
$\bar{\mathsf{R}}$	Maximum service rate to any queue in a time-slot
$h\|_j$	Channel state h restricted to the activated BSs in j
$\mathcal{R}(j,h) \subseteq \mathbb{R}^{M\times n}$	Set of all possible rate vectors for activation vector j and channel state h
$\mathbf{J}(t) = (J_m(t))$	Activation vector at time t
$\mathcal{J}$	Set of all possible activation states
$\mathbf{S}(t) = (S_{m,u}(t))$	Rate allocation at time t
C_1	Cost of operating a BS in ON state
C_0	Cost of switching a BS from ON to OFF state
$C(t)$	Network cost at time t
$Q_{m,u}(t)$	Queue of user u at BS m at the beginning of time-slot t
$\mathcal{P}_l$	Set of all probability (row) vectors in $\mathbb{R}^l$
$\mathcal{P}_l^2$	Set of all stochastic matrices in $\mathbb{R}^{l\times l}$
$\mathcal{W}_l$	Set of all stochastic matrices in $\mathbb{R}^{l\times l}$ with a single ergodic class
$\mathbf{1}_l$	All 1's Column vector of size l
$\mathbf{I}_l$	Identity matrix of size l

(i) Network Cost: We assume that the cost of operating a BS in the *OFF* state (sleep mode) is zero. However, it is easy to include an additional parameter, say C_1', which denotes the cost of a BS in the *OFF* state. Similarly, for switching cost, although we consider only the cost of switching a BS from *ON* to *OFF* state, we can also include the cost of switching from *OFF* to *ON* state (say C_0'). The analysis in this chapter can then be extended by defining the network cost as

$$\begin{aligned} C(t) = \mathsf{C}_0\|(\mathbf{J}(t-1) - \mathbf{J}(t))^+\|_1 + \mathsf{C}_1\|\mathbf{J}(t)\|_1 \\ + \mathsf{C}_0'\|(\mathbf{J}(t) - \mathbf{J}(t-1))^+\|_1 + \mathsf{C}_1'\left(M - \|\mathbf{J}(t)\|_1\right) \end{aligned}$$

instead of (4.1).

(ii) Switching Hysteresis Time: While our system allows switching decisions in every time-slot, we will see that the key to our approach is a slowing of activation set switching dynamics. Specifically, on average our algorithm switches activation states once every $1/\epsilon_s$ timeslots, where ϵ_s is a tunable parameter. Additionally, it is easy to incorporate "hard constraints" on the hysteresis time by restricting the frequency of switching decisions to, say once in every L time-slots (for some constant L). This avoids the problem of switching too frequently and gives a method to implement time-scale separation between the channel allocation decisions and BS activation decisions. While our current algorithm has inter-switching times i.i.d. geometric with mean $1/\epsilon_s$, it is easy to allow other distributions that have bounded means with some independence conditions (independent of each other and also the arrivals and the channel). We skip details in the proofs for notational clarity.

4.3 Optimization Framework

A policy is given by a (possibly random) sequence of resource allocation decisions $(\mathbf{J}(t), \mathbf{S}(t))_{t>0}$. Let $(\mathbf{J}(t-1), \mathbf{Q}(t))$ be the network state at time t.

Notation We use $\mathbb{P}_\varphi[\cdot]$ and $\mathbb{E}_\varphi[\cdot]$ to denote probabilities and expectation under policy φ. We skip the subscript when the policy is clear from the context.

4.3.1 Stability, Network Cost, and the Optimization Problem

Definition 4.1 (Stability)**.** *A network is said to be* stable *under a policy φ if there exist constants $\bar{Q}$, $\rho > 0$ such that for any initial condition $(\mathbf{J}(0), \mathbf{Q}(1))$,*

$$\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{P}_\varphi \left[\sum_{m\in[m],u\in[n]} Q_{m,u}(t) \leq \bar{Q} \,\middle|\, \mathbf{J}(0), \mathbf{Q}(1) \right] > \rho.$$

This definition is motivated by Foster's theorem: indeed, for an aperiodic and irreducible DTMC, Definition 4.1 implies positive recurrence. Consider the set of all ergodic Markov policies $\mathfrak{M}$, including those that know the arrival and channel statistics. A policy $\varphi \in \mathfrak{M}$ makes allocation decisions at time t based only on the current state $(\mathbf{J}(t-1), \mathbf{Q}(t))$ (and possibly the arrival and channel statistical parameters). We now define the support region of a policy and the capacity region.

Definition 4.2 (Support Region of a Policy φ)**.** *The support region $\Lambda^\varphi(\boldsymbol{\mu})$ of a policy φ is the set of all arrival rate vectors for which the network is stable under the policy φ.*

Definition 4.3 (Capacity Region)**.** *The capacity region $\Lambda(\boldsymbol{\mu})$ is the set of all arrival rate vectors for which the network is stable under some policy in $\mathfrak{M}$, i.e., $\Lambda(\boldsymbol{\mu}) :=$*

$\bigcup_{\varphi \in \mathfrak{M}} \Lambda^{\varphi}(\boldsymbol{\mu})$.

Definition 4.4 (Network Cost of a Policy φ)**.** *The network cost $C^{\varphi}(\boldsymbol{\mu}, \boldsymbol{\lambda})$ under a policy φ is the long term average network cost (BS switching and activation costs) per time-slot, i.e.,*

$$C^{\varphi}(\boldsymbol{\mu}, \boldsymbol{\lambda}) := \limsup_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\varphi} \left[C(t) \,\middle|\, \mathbf{J}(0), \mathbf{Q}(1) \right].$$

We formulate the resource allocation problem in a network cost minimization framework. Consider the problem of network cost minimization under Markov policies $\mathfrak{M}$ subject to stability. The optimal network cost is given by

$$C^{\mathfrak{M}}(\boldsymbol{\mu}, \boldsymbol{\lambda}) := \inf_{\{\varphi \in \mathfrak{M} : \lambda \in \Lambda^{\varphi}(\boldsymbol{\mu})\}} C^{\varphi}(\boldsymbol{\mu}, \boldsymbol{\lambda}). \tag{4.2}$$

4.3.2 Markov-Static-Split Rules

The capacity region $\Lambda(\boldsymbol{\mu})$ will naturally be characterized by only those Markov policies that maintain all the BSs active in all the time-slots, i.e., $\mathbf{J}(t) = \mathbf{1}\, \forall t$. In the traditional scheduling problem without BS switching, it is well-known that the capacity region can be characterized by the class of *static-split* policies [15] that allocate rates in a random i.i.d. fashion given the current channel state. An arrival rate vector $\boldsymbol{\lambda} \in \Lambda(\boldsymbol{\mu})$ iff there exists convex combinations $\left\{ \boldsymbol{\alpha}(\mathbf{1}, h) \in \mathcal{P}_{|\mathcal{R}(\mathbf{1},h)|} \right\}_{h \in \mathcal{H}}$ such that

$$\boldsymbol{\lambda} < \sum_{h \in \mathcal{H}} \mu(h) \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{1},h)} \alpha_{\mathbf{r}}(\mathbf{1}, h) \mathbf{r}.$$

But note that static-split rules in the above class, in which BSs are not switched *OFF*, do not optimize the network cost.

We now describe a class of activation policies called the *Markov-static-split + static-split* rules which are useful in handling the network cost. A policy is a Markov-static-split + static-split rule if it uses a time-homogeneous Markov rule for BS activation in every time-slot and an i.i.d. static-split rule for rate allocations. For any $l \in \mathbb{N}$, let $\mathcal{W}_l$ denote the set of all stochastic matrices of size l with a single ergodic class. A Markov-static-split + static-split rule is characterized by

1. a stochastic matrix $\mathbf{P} \in \mathcal{W}_{|\mathcal{J}|}$ with a single ergodic class,
2. convex combinations $\left\{\boldsymbol{\alpha}(j,h) \in \mathcal{P}_{|\mathcal{R}(j,h)|}\right\}_{j\in\mathcal{J},h\in\mathcal{H}}$.

Here $\mathbf{P}$ represents the transition probability matrix that specifies the jump probabilities from one activation state to another in successive time-slots. $\{\boldsymbol{\alpha}(j,h)\}_{j\in\mathcal{J},h\in\mathcal{H}}$ specify the static-split rate allocation policy given the activation state and the network channel-state.

Let $\mathfrak{MS}$ denote the class of all Markov-static-split + static-split rules. For a rule $\left(\mathbf{P}, \boldsymbol{\alpha} = \{\boldsymbol{\alpha}(j,h)\}_{j\in\mathcal{J},h\in\mathcal{H}}\right) \in \mathfrak{MS}$, let $\boldsymbol{\sigma}$ denote the invariant probability distribution corresponding to the stochastic matrix $\mathbf{P}$. Then the expected switching and activation costs are given by $\mathsf{C}_0 \sum_{j',j\in\mathcal{J}} \sigma_{j'} P_{j',j} \|(j'-j)^+\|_1$ and $\mathsf{C}_1 \sum_{j\in\mathcal{J}} \sigma_j \|j\|_1$ respectively. We prove in the following theorem that the class $\mathfrak{MS}$ can achieve the same performance as $\mathfrak{M}$, the class of all ergodic Markov policies.

Theorem 4.1. *For any $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ and $\varphi \in \mathfrak{M}$ such that $\boldsymbol{\lambda} \in \Lambda^{\varphi}(\boldsymbol{\mu})$, there exists a $\varphi' \in \mathfrak{MS}$ such that $\boldsymbol{\lambda} \in \Lambda^{\varphi'}(\boldsymbol{\mu})$ and $C^{\varphi'}(\boldsymbol{\mu},\boldsymbol{\lambda}) = C^{\varphi}(\boldsymbol{\mu},\boldsymbol{\lambda})$. Therefore,*

$$C^{\mathfrak{M}}(\boldsymbol{\mu},\boldsymbol{\lambda}) = \inf_{\varphi'\in\mathfrak{MS},\lambda\in\Lambda^{\varphi'}(\boldsymbol{\mu})} C^{\varphi'}(\boldsymbol{\mu},\boldsymbol{\lambda}).$$

Proof Outline. The proof of this theorem is similar to the proof of characterization of the stability region using the class of static-split policies. It maps the time-averages of BS activation transitions and rate allocations of the policy $\varphi \in \mathfrak{M}$ to a Markov-static-split rule $\varphi' \in \mathfrak{MS}$ that mimics the same time-averages. For complete proof, please see Appendix C. □

From the characterization of the class $\mathfrak{MS}$, Theorem 4.1 shows that $C^{\mathfrak{M}}(\boldsymbol{\mu}, \boldsymbol{\lambda})$ is equal to the optimal value, $V(\mu, \lambda)$, of the optimization problem given below.

$$\inf_{\mathbf{P},\boldsymbol{\alpha}} \mathsf{C}_0 \sum_{j',j\in\mathcal{J}} \sigma_{j'} P_{j',j} \|\left(j'-j\right)^+\|_1 + \mathsf{C}_1 \sum_{j\in\mathcal{J}} \sigma_j \|j\|_1$$

such that $\mathbf{P} \in \mathcal{W}_{|\mathcal{J}|}$ with unique invariant distribution $\boldsymbol{\sigma} \in \mathcal{P}_{|\mathcal{J}|}$, and $\boldsymbol{\alpha}(j,h) \in \mathcal{P}_{|\mathcal{R}(j,h)|}\, \forall j \in \mathcal{J}, h \in \mathcal{H}$ with

$$\boldsymbol{\lambda} < \sum_{j\in\mathcal{J}} \sigma_j \sum_{h\in\mathcal{H}} \mu(h) \sum_{\mathbf{r}\in\mathcal{R}(j,h)} \alpha_{\mathbf{r}}(j,h)\mathbf{r}. \tag{4.3}$$

4.3.3 A Modified Optimization Problem

Now, consider the linear program given by

$$\min_{\boldsymbol{\sigma},\boldsymbol{\beta}} \mathsf{C}_1 \sum_{j\in\mathcal{J}} \sigma_j \|j\|_1, \quad \text{such that}$$

$$\begin{aligned}
\boldsymbol{\sigma} &\in \mathcal{P}_{|\mathcal{J}|} \\
\beta_{j,h,\mathbf{r}} &\geq 0 \quad \forall \mathbf{r} \in \mathcal{R}(j,h), \forall j \in \mathcal{J}, h \in \mathcal{H}, \\
\sigma_j &= \sum_{\mathbf{r}\in\mathcal{R}(j,h)} \beta_{j,h,\mathbf{r}} \quad \forall j \in \mathcal{J}, h \in \mathcal{H}, \\
\boldsymbol{\lambda} &\leq \sum_{\substack{j\in\mathcal{J},h\in\mathcal{H},\\ \mathbf{r}\in\mathcal{R}(j,h)}} \beta_{j,h,\mathbf{r}} \mu(h)\mathbf{r}.
\end{aligned} \tag{4.4}$$

Let $d := |\mathcal{J}| + \sum_{j\in\mathcal{J},h\in\mathcal{H}}|\mathcal{R}(j,h)|$ be the number of variables in the above linear program. We denote by $L_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$, a linear program with constraints as above and with $\mathbf{c} \in \mathbb{R}^d$ as the vector of weights in the objective function. Thus, the feasible set of the linear program $L_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$ is specified by the parameters $\boldsymbol{\mu},\boldsymbol{\lambda}$ and the objective function is specified by the vector $\mathbf{c}$. Let $C^*_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$ denote the optimal value of $L_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$ and $\mathcal{O}^*_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$ denote the optimal solution set. Also, let

$$\mathcal{S} := \{(\boldsymbol{\mu},\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in \Lambda(\boldsymbol{\mu})\},$$

$$\mathcal{U}_{\mathbf{c}} := \{(\boldsymbol{\mu},\boldsymbol{\lambda}) \in \mathcal{S} : L_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda}) \text{ has a unique solution}\}.$$

Note that $L_{\mathbf{c}^0}(\boldsymbol{\mu},\boldsymbol{\lambda})$, with

$$\mathbf{c}^0 := ((\mathsf{C}_1\|j\|_1)_{j\in\mathcal{J}}, \mathbf{0}) \tag{4.5}$$

is a relaxation of the original optimization problem $V(\boldsymbol{\mu},\boldsymbol{\lambda})$, and therefore

$$C^*_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda}) \leq C^{\mathfrak{M}}(\boldsymbol{\mu},\boldsymbol{\lambda}). \tag{4.6}$$

We use results from [139], [44] to show (in the Lemma below) that the solution set and the optimal value of the linear program are continuous functions of the input parameters.

Lemma 4.1. *(I) As a function of the weight vector* $\mathbf{c}$ *and the parameters* $\boldsymbol{\mu},\boldsymbol{\lambda}$*, the optimal value* $C^*_{(\cdot)}(\cdot)$ *is continuous at any* $(\mathbf{c},(\boldsymbol{\mu},\boldsymbol{\lambda})) \in \mathbb{R}^d \times \mathcal{S}$*.*

(II) For any weight vector $\mathbf{c}$*, the optimal solution set* $\mathcal{O}^*_{\mathbf{c}}(\cdot)$*, as a function of the parameters* $(\boldsymbol{\mu},\boldsymbol{\lambda})$*, is continuous at any* $(\boldsymbol{\mu},\boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}}$*.*

Remark 4.1. *Since* $\mathcal{O}^*_{\mathbf{c}}(\boldsymbol{\mu},\boldsymbol{\lambda})$ *is a singleton if* $(\boldsymbol{\mu},\boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}}$*, the definition of continuity in this context is unambiguous.*

4.3.4 A Feasible Solution: Static-Split + Max-Weight

We now discuss how we can use the linear program L to obtain a feasible solution for the original optimization problem (**??**). We need to deal with two modified constraints:

(i) Single Ergodic Class – Spectral Gap: For any $\boldsymbol{\sigma} \in \mathcal{P}_{|\mathcal{J}|}$ and $\epsilon_s \in (0,1)$, the stochastic matrix

$$\mathbf{P}(\boldsymbol{\sigma}, \epsilon_s) := \epsilon_s \mathbf{1}_{|\mathcal{J}|}\boldsymbol{\sigma} + (1-\epsilon_s)\mathbf{I}_{|\mathcal{J}|} \tag{4.7}$$

is aperiodic and has a single ergodic class given by $\{j : \sigma_j > 0\}$. Therefore, given any optimal solution $(\boldsymbol{\sigma}, \boldsymbol{\beta})$ for the relaxed problem $L_{\mathbf{c}}(\boldsymbol{\mu}, \boldsymbol{\lambda})$, we can construct a feasible solution $(\mathbf{P}(\boldsymbol{\sigma}, \epsilon_s), \boldsymbol{\alpha})$ for the original optimization problem $V(\boldsymbol{\mu}, \boldsymbol{\lambda})$ such that the network cost for this solution is at most $\epsilon_s M \mathsf{C}_0$ more than the optimal cost. Note that ϵ_s is the spectral gap of the matrix $\mathbf{P}(\boldsymbol{\sigma}, \epsilon_s)$.

(ii) Stability – Capacity Gap: To ensure stability, it is necessary that the arrival rate is strictly less than the service rate (inequality (4.3)). It can be shown that an optimal solution to the linear program satisfies the constraint (4.4) with equality, and therefore cannot guarantee stability. An easy solution to this problem is to solve a modified linear program with a fixed small gap ϵ_g between the arrival rate and the offered service rate. We refer to the parameter ϵ_g as the *capacity gap*. Continuity of the optimal cost of the linear program L (from part (I) of Lemma 4.1) ensures that the optimal cost of the modified linear program is close to the optimal cost of the original optimization problem for sufficiently small ϵ_g.

To summarize, if the statistical parameters $\boldsymbol{\mu}, \boldsymbol{\lambda}$ were known, one could adopt the following scheduling policy:

(a) BS activation: Compute an optimal solution $(\boldsymbol{\sigma}^*, \boldsymbol{\beta}^*)$ for the linear program $L_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$. Initially, choose a BS state according to the static-split rule given by $\boldsymbol{\sigma}^*$. Subsequently, at every time-slot, with probability $1 - \epsilon_s$, maintain the BSs in the same state as the previous time-slot, i.e., no switching. With probability ϵ_s, choose a new BS state according to the static-split rule given by $\boldsymbol{\sigma}^*$. The network can be operated at a cost close to the optimal by choosing ϵ_s, ϵ_g sufficiently small.

(b) Rate allocation: To ensure stability, use a queue-based rule such as the Max-Weight rule to allocate rates given the observed channel state:

$$\mathbf{S}(t) = \underset{\mathbf{r} \in \mathcal{R}(\mathbf{J}(t), H(t))}{\arg\max} \mathbf{Q}(t) \cdot \mathbf{r}. \tag{4.8}$$

We denote the above static-split + Max-Weight rule with parameters ϵ_s, ϵ_g by $\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g, \epsilon_s)$. The theorem below shows that the static-split + Max-Weight policy achieves close to optimal cost while ensuring queue stability.

Theorem 4.2. *For any $\boldsymbol{\mu}, \boldsymbol{\lambda}$ such that $(\boldsymbol{\mu}, \boldsymbol{\lambda} + 2\epsilon_g) \in \mathcal{S}$, and for any $\epsilon_s \in (0, 1)$, under the static-split + Max-Weight rule $\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g, \epsilon_s)$,*

1. *the network cost satisfies*

$$C^{\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g, \epsilon_s)}(\boldsymbol{\mu}, \boldsymbol{\lambda}) \leq C^{\mathfrak{M}}(\boldsymbol{\mu}, \boldsymbol{\lambda}) + \kappa \epsilon_s + \gamma(\epsilon_g),$$

for some constant κ that depends on the network size and C_0, C_1, and for some increasing function $\gamma(\cdot)$ such that $\lim_{\epsilon_g \to 0} \gamma(\epsilon_g) = 0$, and

2. the network is stable, i.e.,

$$\boldsymbol{\lambda} \in \Lambda^{\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g, \epsilon_s)}(\boldsymbol{\mu}).$$

Proof Outline. Since $\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)$ has a single ergodic class, the marginal distribution of the activation state $(\mathbf{J}(t))_{t>0}$ converges to $\boldsymbol{\sigma}^*$. Part 1 of the theorem then follows from (4.6) and the continuity of the optimal value of L (Lemma 4.1(I)). Part 2 relies on the strict inequality gap enforced by ϵ_g in (4.3). Therefore, it is possible to serve all the arrivals in the long-term. We use a standard Lyapunov argument which shows that the T-step quadratic Lyapunov drift for the queues is strictly negative outside a finite set for some $T > 0$. For complete proof, please see Appendix C. □

One can also achieve the above guarantees with a static-split + static-split rule which has BS activations as above but channel allocation through a static-split rule with convex combinations given by $\boldsymbol{\alpha}^*$ such that

$$\alpha^*_{\mathbf{r}}(j,h) = \frac{\beta^*_{j,h,\mathbf{r}}}{\sigma^*_j} \quad \forall \mathbf{r} \in \mathcal{R}(j,h), \forall j \in \mathcal{J}, h \in \mathcal{H}. \tag{4.9}$$

4.3.5 Effect of Parameter Choice on Performance

ϵ_s and ϵ_g can be used as a control parameters to trade-off between two desirable but conflicting features – small queue lengths and low network cost.

(i) Spectral gap, ϵ_s: ϵ_s is the spectral gap of the transition probability matrix $\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)$ and, therefore, impacts the mixing time of the activation state $(\mathbf{J}(t))_{t>0}$. Since the average available service rate is dependent on the distribution of the activation state, the time taken for the queues to stabilize depends on the mixing

time, and consequently, on the choice of ϵ_s. With $\epsilon_s = 1$, we are effectively ignoring switching costs as this corresponds to a rule that chooses the activation sets in an i.i.d. manner according to the distribution $\boldsymbol{\sigma}^*$. Thus, stability is ensured but at a penalty of larger average costs. At the other extreme, when $\epsilon_s = 0$, the transition probability matrix $\mathbf{I}_{|\mathcal{J}|}$ corresponds to an activation rule that never switches the BSs from their initial activation state. This extreme naturally achieves zero switching cost, but at the cost of queue stability as the activation set is frozen for all times.

(ii) Capacity gap, ϵ_g: Recall that ϵ_g is the gap enforced between the arrival rate and the allocated service rate in the linear program $L_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$. Since the mean queue-length is known to vary inversely as the capacity gap, the parameter ϵ_g can be used to control queue-lengths. A small ϵ_g results in low network cost and large mean queue-lengths.

4.4 Policy with Unknown Statistics

In the setting where arrival and channel statistics are unknown, our interest is in designing policies that learn the arrival and channel statistics to make rate allocation and BS activation decisions. As described in Section 4.2.2, channel rates are observed in every time-slot after activation of the BSs. Since only channel rates of activated BSs can be obtained in any time-slot, the problem naturally involves a trade-off between activating more BSs to get better channel estimates versus maintaining low network cost. Our objective is to design policies that achieve network cost close to $C^{\mathfrak{M}}$ while learning the statistics well enough to stabilize the queues.

4.4.1 An Explore-Exploit Policy

Algorithm 3 Policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$ with parameters ϵ_p, ϵ_s, ϵ_g

Generate a uniformly distributed random direction $\boldsymbol{\upsilon} \in \mathbb{R}^d$.
Construct a perturbed weight vector

$$\mathbf{c}^{\epsilon_p} \leftarrow \mathbf{c}^0 + \epsilon_p \boldsymbol{\upsilon}.$$

Initialize $\hat{\boldsymbol{\mu}} \leftarrow \mathbf{0}$, $\hat{\boldsymbol{\lambda}} \leftarrow \mathbf{0}$.
for all $t > 0$ **do**
    Generate $E_l(t)$, an independent Bernoulli sample of mean $\epsilon_l(t) = \frac{2 \log t}{t}$.
    if $E_l(t) = 1$ **then** ▷ *Explore*
        $\mathbf{J}(t) \leftarrow \mathbf{1}$ (Activate all the BSs).
        Observe the channel state $H(t)$.
        Update empirical distributions $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\lambda}}$.
    else ▷ *Exploit*
        Generate $E_s(t)$, an independent Bernoulli sample of mean ϵ_s.
        if $E_s(t) = 0$ **then** ▷ *No Switching*
            $\mathbf{J}(t) \leftarrow \mathbf{J}(t-1)$.
        else
            Solve $L_{\mathbf{c}^{\epsilon_p}}\left(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}} + \epsilon_g\right)$ and select an optimal solution $\left(\hat{\boldsymbol{\sigma}}(t), \hat{\boldsymbol{\beta}}(t)\right)$.
            Select $\mathbf{J}(t)$ according to the distribution $\hat{\boldsymbol{\sigma}}(t)$.
        end if
        Observe the channel state $H(t)|_{\mathbf{J}(t)}$.
    end if
    Allocate channels according to the Max-Weight Rule,

$$\mathbf{S}(t) \leftarrow \underset{\mathbf{r} \in \mathcal{R}(\mathbf{J}(t), H(t))}{\arg\max} \mathbf{Q}(t) \cdot \mathbf{r}.$$

end for

Algorithm 3 gives a policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$, which is an explore-exploit strategy similar to the ϵ-greedy policy in the multi-armed bandit problem. Here, $\epsilon_p, \epsilon_s, \epsilon_g$ are fixed parameters of the policy.

4.4.1.1 Initial Perturbation of the Cost Vector

Given the original cost vector $\mathbf{c}^0$ (given by (4.5)), the policy first generates a slightly perturbed cost vector $\mathbf{c}^{\epsilon_p}$ by adding to $\mathbf{c}^0$, a random perturbation uniformly distributed on the ϵ_p-ball. It is easily verified that, for any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$,

$$|C^*_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda}) - C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda})| \leq \sqrt{|\mathcal{H}| + 1}\mathsf{C}_1\epsilon_p.$$

In addition, the following lemma shows that the perturbed linear program has a unique solution with probability 1.

Lemma 4.2. *For any* $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$,

$$\mathbb{P}\left[(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}^{\epsilon_p}} \mid \mathbf{J}(0), \mathbf{Q}(1)\right] = 1.$$

4.4.1.2 BS Activation

At any time t the policy randomly chooses to *explore* or *exploit*. The probability that it explores, $\epsilon_l(t) = \frac{2\log t}{t}$, decreases with time.

Exploration. In the *explore* phase, the policy activates all the BSs and observes the channel. It maintains $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}}$, the empirical distribution of the channel and the empirical mean of the arrival vector respectively, obtained from samples in the explore phase.

Exploitation. In the *exploit* phase, with probability $1 - \epsilon_s$, the policy chooses to keep the same activation set as the previous time-slot (i.e., no switching). With probability ϵ_s, it solves the linear program $L_{\mathbf{c}^{\epsilon_p}}\left(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}} + \epsilon_g\right)$ with the perturbed cost vector $\mathbf{c}^{\epsilon_p}$ and parameters $\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}} + \epsilon_g$ given by the empirical distribution. From

an optimal solution $\left(\hat{\boldsymbol{\sigma}}(t), \hat{\boldsymbol{\beta}}(t)\right)$ of the linear program, it chooses the BS activation vector $\mathbf{J}(t)$ according to the distribution $\hat{\boldsymbol{\sigma}}(t)$.

4.4.1.3 Rate Allocation

The policy uses the Max-Weight Rule given by (4.8) for channel allocation.

4.4.2 Performance Guarantees

In Theorem 4.3, we give stability and network cost guarantees for the proposed learning-cum-scheduling rule $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$.

Theorem 4.3. *For any $\boldsymbol{\mu}, \boldsymbol{\lambda}$ such that $(\boldsymbol{\mu}, \boldsymbol{\lambda} + 2\epsilon_g) \in \mathcal{S}$, and for any $\epsilon_p, \epsilon_s \in (0,1)$, under the policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$,*

1. *the network cost satisfies*

$$C^{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}(\boldsymbol{\mu}, \boldsymbol{\lambda}) \leq C^{\mathfrak{M}}(\boldsymbol{\mu}, \boldsymbol{\lambda}) + \kappa(\epsilon_p + \epsilon_s) + \gamma(\epsilon_g),$$

   for some constant κ that depends on the network size and C_0, C_1, and for some increasing function $\gamma(\cdot)$ such that $\lim_{\epsilon_g \to 0} \gamma(\epsilon_g) = 0$, and

2. *the network is stable, i.e.,*

$$\boldsymbol{\lambda} \in \Lambda^{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}(\boldsymbol{\mu}).$$

Proof Outline. As opposed to known statistical parameters for the arrivals and the channel in the Markov-static-split rule, the policy uses empirical statistics that change dynamically with time. Thus, the activation state process $(\mathbf{J}(t))_{t>0}$, in this case, is not a time-homogeneous Markov chain. However, we note that $\mathbf{J}(t)$ along

with the empirical statistics forms a time-inhomogeneous Markov chain with the empirical statistics converging to the true statistics almost surely. Specifically, we show that the time taken by the algorithm to learn the parameters within a small error has a finite second moment.

We then use convergence results for time-inhomogeneous Markov chains (derived in Lemma 4.3 in Section 4.5) to show convergence of the marginal distribution of the activation state $(\mathbf{J}(t))_{t>0}$. As in Theorem 4.2, Part 1 then follows from (4.6) and the continuity of the optimal value of L (Lemma 4.1(I)).

Part 2 requires further arguments. The queues have a negative Lyapunov drift only after the empirical estimates have converged to the true parameters within a small error. To bound the Lyapunov drift before this time, we use boundedness of the arrivals along with the existence of second moment for the convergence time of the estimated parameters. By using a telescoping argument as in Foster's theorem, we show that this implies stability as per Definition 4.1. For complete proof, please see Appendix C. □

4.4.3 Discussion: Other Potential Approaches

Recall that our system consists of two distinct time-scales: (a) exogenous fast dynamics due the channel variability, that occurs on a per-time-slot basis, and (b) endogenous slow dynamics of learning and activation due to base-station active-sleep state dynamics. By 'exogenous', we mean that the time-scale is controlled by nature (channel process), and by 'endogenous', we mean that the time-scale is controlled by the learning-cum-activation algorithm (slowed dynamics where activation

states change only infrequently). To place this in perspective, consider the following alternate approaches, each of which has defects.

1. Virtual queues + MaxWeight: As is now standard [56, 127], suppose that we encode the various costs through virtual queues (or variants there-of), and apply a MaxWeight algorithm to this collection of queues. Due to the switching cost, effective channel – the vector of channel rates on the active collection of base-stations – has dependence across time (coupled dynamics of channel and queues) through the activation set scheduling, and voids the standard Lyapunov proof approach for showing stability. Specifically, we cannot guarantee that the time average of various activation sets chosen by this (virtual + actual queue) MaxWeight algorithm equals the corresponding optimal fractions computed using a linear program with known channel and arrival parameters.

2. Ignoring Switching Costs with Fast Dynamics: Suppose we use virtual queues to capture only the activation costs. In this case, a MaxWeight approach (selecting a new activation set and channel allocation in each time-slot) will ensure stability, but will not provide any guarantees on cost optimality as there will be frequent switching of the activation set.

3. Ignoring Switching Costs with Slowed Dynamics: Again, we use virtual queues for encoding only activation costs, and use block scheduling. In other words, re-compute an activation + channel schedule once every R time-slots, and use this fixed schedule for this block of time (pick-and-compare, periodic, frame-based algorithms [130, 108, 40, 145]). While this approach minimizes switching costs (as activation changes occur infrequently), stability properties are lost as we are not making

use of opportunism arising from the wireless channel variability (the schedule is fixed for a block of time and does not adapt to instantaneous channel variations).

Our approach avoids the difficulties in each of these approaches by explicitly slowing down the time-scale of the activation set dynamics (an engineered slow time-scale), thus minimizing switching costs. However, it allows channels to be opportunistically re-allocated in each time-slot based on the instantaneous channel state (the fast time-scale of nature). This fast-slow co-evolution of learning, activation sets and queue lengths requires a new proof approach. We combine new results (see Section 4.5) on convergence of inhomogeneous Markov chains with Lyapunov analysis to show both stability and cost (near) optimality.

4.5 Convergence of a Time-Inhomogeneous Markov Process

In the following section, we derive some convergence bounds for perturbed time-inhomogeneous Markov chains which are useful in proving stability and cost optimality. Let $\mathcal{P} := \{\mathbf{P}_\delta, \delta \in \Delta\}$ be a collection of stochastic matrices in $\mathbb{R}^{N\times N}$, with $\{\boldsymbol{\sigma}_\delta, \delta \in \Delta\}$ denoting the corresponding invariant probability distributions. Also, let $\mathbf{P}_*$ be an $N \times N$ aperiodic stochastic matrix with a single ergodic class and invariant probability distribution $\boldsymbol{\sigma}_*$.

Recall that for a stochastic matrix $\mathbf{P}$ the coefficient of ergodicity [124] $\tau_1(\mathbf{P})$ is defined by

$$\tau_1(\mathbf{P}) := \max_{\mathbf{z}^\mathsf{T}\mathbf{1}_N=0\,,\,\|\mathbf{z}\|_1=1} \|\mathbf{P}^\mathsf{T}\mathbf{z}\|_1. \tag{4.10}$$

It has the following basic properties [124]:

1. $\tau_1(\mathbf{P}_1\mathbf{P}_2) \leq \tau_1(\mathbf{P}_1)\tau_1(\mathbf{P}_2)$,

2. $|\tau_1(\mathbf{P}_1) - \tau_1(\mathbf{P}_2)| \leq \|\mathbf{P}_1 - \mathbf{P}_2\|_\infty$,

3. $\|\mathbf{xP} - \mathbf{yP}\|_1 \leq \tau_1(\mathbf{P}) \|\mathbf{x} - \mathbf{y}\|_1 \quad \forall\, \mathbf{x}, \mathbf{y} \in \mathcal{P}_N$, and

4. $\tau_1(\mathbf{P}) < 1$ if and only if $\mathbf{P}$ has no pair of orthogonal rows (i.e., if it is a scrambling matrix).

By the results in [16], if $\mathbf{P}_*$ is aperiodic and has a single ergodic class then there exists an integer $\hat{m}$ such that $\mathbf{P}_*^k$ is scrambling for all $k \geq \hat{m}$. Therefore, $\tau_1(\mathbf{P}_*^k) < 1\ \forall k \geq \hat{m}$.

Define

$$\epsilon := \sup_{\delta \in \Delta} \|\mathbf{P}_\delta - \mathbf{P}_*\|_1. \tag{4.11}$$

Now, consider a time-inhomogeneous Markov chain $(X(t))_{t\geq 0}$ with initial distribution $\mathbf{y}(0)$, and transition probability matrix at time t given by $\mathbf{P}_{\delta_t} \in \mathcal{P}\ \forall t > 0$. Let $\{\mathbf{y}(t)\}_{t\geq 0}$ be the resulting sequence of marginal distributions. The following lemma gives a bound on the convergence of the limiting distribution of such a time-inhomogeneous DTMC to $\boldsymbol{\sigma}_*$. Additional results are available in the Appendix.

Lemma 4.3. *For any* $\mathbf{y}(0)$*,*

(a) the marginal distribution satisfies

$$\|\mathbf{y}(n) - \boldsymbol{\sigma}_*\|_1 \leq \tau_1(\mathbf{P}_*^n)\|\mathbf{y}(0) - \boldsymbol{\sigma}_*\|_1 + \epsilon \sum_{\ell=0}^{n-1} \tau_1(\mathbf{P}_*^\ell)\,, \tag{4.12}$$

(b) and the limiting distribution satisfies

$$\limsup_{n\to\infty} \|\mathbf{y}(n) - \boldsymbol{\sigma}_*\|_1 \;\le\; \epsilon\, \Upsilon(\mathbf{P}_*)$$

where $\Upsilon(\mathbf{P}_*) := \sum_{\ell=0}^{\infty} \tau_1(\mathbf{P}_*^\ell) \;\le \frac{\hat{m}}{1-\tau_1(\mathbf{P}_*^{\hat{m}})}$.

Proof. The trajectory $(\mathbf{y}(n))_{n>0}$ satisfies $\forall n \ge 1$,

$$\mathbf{y}(n) \;=\; \mathbf{y}(n-1)\mathbf{P}_* + \mathbf{y}(n-1)(\mathbf{P}_{\delta_{n-1}} - \mathbf{P}_*). \tag{4.13}$$

Using (4.13) recursively, we have

$$\mathbf{y}(n) \;=\; \mathbf{y}(0)\mathbf{P}_*^n + \sum_{k=1}^{n} \mathbf{y}(n-k)(\mathbf{P}_{\delta_{n-k}} - \mathbf{P}_*)\mathbf{P}_*^{k-1},$$

which gives us

$$\begin{aligned}\mathbf{y}(n) - \boldsymbol{\sigma}_* = (\mathbf{y}(0) - \boldsymbol{\sigma}_*)\mathbf{P}_*^n \\ + \sum_{k=1}^{n} \mathbf{y}(n-k)(\mathbf{P}_{\delta_{n-k}} - \mathbf{P}_*)\mathbf{P}_*^{k-1}.\end{aligned} \tag{4.14}$$

Now, taking norms and using the definitions in (4.10) and (4.11), we obtain

$$\|\mathbf{y}(n) - \boldsymbol{\sigma}_*\|_1 \;\le\; \tau_1(\mathbf{P}_*^n)\|\mathbf{y}(0) - \boldsymbol{\sigma}_*\|_1 + \epsilon \sum_{\ell=0}^{n-1} \tau_1(\mathbf{P}_*^\ell)\,.$$

This proves part (a) of the lemma. Now, note that

$$\tau_1(\mathbf{P}_*^k) \;\le\; \big(\tau_1(\mathbf{P}_*^m)\big)^{\lfloor k/m \rfloor} \tag{4.15}$$

for any positive integers k, m. Since $\tau_1(\mathbf{P}_*^{\hat{m}}) < 1$, it follows that $\lim_{n\to\infty} \tau_1(\mathbf{P}_*^n) = 0$, and

$$\Upsilon(\mathbf{P}_*) \;=\; \sum_{\ell=0}^{\infty} \tau_1(\mathbf{P}_*^\ell) \;\le \frac{\hat{m}}{1-\tau_1(\mathbf{P}_*^{\hat{m}})}.$$

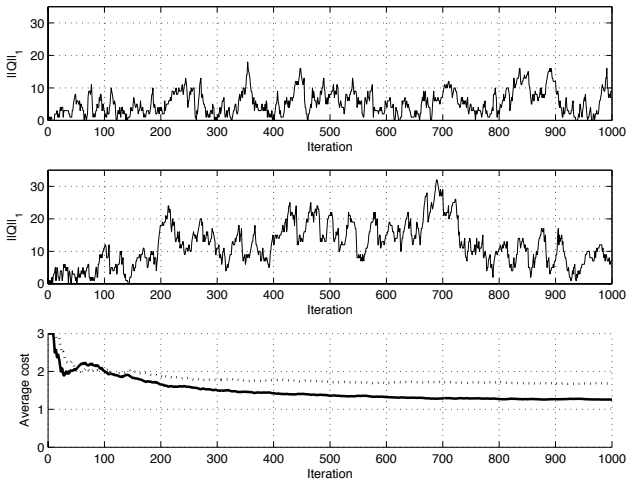


Figure 4.1: The top two plots show the total queue size as a function of time when $\epsilon_s = 0.2$ and $\epsilon_s = 0.05$, respectively. The bottom plot shows the corresponding average costs (with the solid curve for $\epsilon_s = 0.05$). A smaller ϵ_s yields a lower average cost but has higher average queue size.

Using this in (4.12), we have

$$\limsup_{n\to\infty} \|\mathbf{y}(n) - \boldsymbol{\sigma}_*\|_1 \leq \epsilon\, \Upsilon(\mathbf{P}_*) \leq \frac{\epsilon\, \hat{m}}{1 - \tau_1(\mathbf{P}_*^{\hat{m}})}\,,$$

which proves part (b) of the lemma. □

4.6 Simulation Results

We present simulations that corroborate the theoretical results in this chapter. The setting is as follows. There are five users and three BSs in the system. BS 1 can service users 1, 2, and 5. BS 2 can service users 1, 2, 3, and 4. BS 3 can service users 3, 4, and 5. The Bernoulli arrival rates on each queue (which have to be learned by the algorithm) is 0.1 packets/slot on each mobile-BS service connection. The total arrival rate to the system is thus 0.1 packet/slot × 10 con-

nections, or 1 packet/slot. A good channel yields a service of 2 packets/slot while a bad channel yields 1 packet/slot. In our correlated fading model, either all channels are bad, or all connections to exactly one BS are good while the others bad. This yields four correlated channel states and all four are equiprobable (the probabilities being unknown to the algorithm). The fading process is independent and identically distributed over time. The activation constraint is that each BS can service at most one mobile per slot. The per BS switching cost C_0 and activation cost C_1 are both taken to be 1. Figure 4.1 provides the average queue sizes (first two plots) and average costs (third plot) for two values of ϵ_s, namely, 0.2 (first plot) and 0.05 (second plot). The plots show that a smaller ϵ_s yields a lower average cost and stabilizes the queue, but has higher average queue size.

4.7 Summary

We study the problem of jointly activating base-stations along with channel allocation, with the objective of minimizing energy costs (activation + switching) subject to packet queue stability. Our approach is based on timescale decomposition, consisting of fast-slow co-evolution of user queues (fast) and base-station activation sets (slow). We develop a learning-cum-scheduling algorithm that can achieve an average cost that is arbitrarily close to optimal, and simultaneously stabilize the user queues. The proposed algorithm is analyzed using novel results in convergence of inhomogeneous Markov chains.

Chapter 5

Conclusion

This thesis deals with the design of algorithms for learning and resource allocation in unknown or partially known environments. We study three such discrete-time control problems:

(i) Detection of sponsored ads in a recommender system,

(ii) Scheduling in a parallel server system with unknown statistics,

(iii) Joint BS activation and rate allocation in a cellular network.

We formulate each of these problems in an online decision-making framework where the controller obtains dynamic feedback that implicitly gives information about the environment as it makes sequential decisions. For all the three problems, we propose solutions in which the feedback data is appropriately used to extract useful information and compute the decision at each time.

This work emphasizes the need for appropriately using implicit feedback data to extract useful information about the environment and balance the gains obtained from exploration and exploitation. For example, in the recommender systems problem, we show how a counting based metric, and not average rating, is appropriate for the inference problem at hand. We note that the queueing bandit problem can be

placed somewhere in between a pure learning problem and the MAB problem, and therefore warrants an exploration strategy which falls in between pure exploration and standard bandit algorithms. We use this insight to incorporate structured exploration in our scheduling algorithm for the queueing-bandit problem. Likewise, in the joint resource allocation problem for energy-efficient networks, we adopt two different strategies of using the implicit data for the two constituent problems – for activating the base-stations, we make switching decisions very infrequently and make use of estimated channel parameters. Whereas rate allocation decisions are made in every time-slot by using the instantaneous channel information.

This thesis also highlights the importance of choosing a performance objective that is appropriate for the application. For example, the traditional multi-armed bandit (MAB) problem (which also falls under the class of sequential decision-making problems) has cumulative reward as the performance objective, and algorithms like UCB1 and Thompson Sampling have been shown to optimal. But many online scheduling applications with a queueing structure require optimization of customer wait times which is better represented by queue-lengths rather than cumulative scheduled service. The queueing-bandit problem, which has the queue-regret as the performance objective, is therefore more appropriate for such applications. Another example which illustrates the significance of choosing the right performance objective is the inclusion of switching cost in the design of energy efficient cellular networks. It is indeed remarkable that the standard Lyapunov approach that yields optimal performance for networks with no switching cost cannot be used for networks with switching cost. These examples show how it is important to adopt an

application-appropriate model.

5.1 Future Directions

There remain substantial open directions for future work, as we briefly discuss.

5.1.1 Detecting Sponsored Recommendations

For this problem, first, it is important to understand the challenges involved in deploying such an algorithm in a practical setting. These could vary depending upon the application. For e.g., detection of pharmaceutical lobbying using this approach requires a modestly large medical database consisting of prescriptions and their efficacy on patients.

It would also be interesting to study extensions of this problem. For e.g., there could be an underlying social graph between the users or a causal relation between recommended items (say, two drugs which are often prescribed one after another) which introduces structural dependence between different recommendations. How important is the knowledge of this structure for the recommender systems and for the anomaly detection system?

5.1.2 Bandit Algorithms for Queueing Systems

As mentioned at the end of Chapter 3, we do not yet know if there is a single algorithm that gives optimal performance in *both* early and late stages. While Q-UCB is asymptotically optimal (to within poly-logarithmic factors) and also es-

sentially exhibits the correct switching behavior between early and late stages, it is not optimal in the early stage: the price paid for structured exploration is an inflation of regret in this stage. An important open question is to find a single, adaptive algorithm that gives good performance over all time. On this point, we note that Thompson sampling [132] is an intriguing candidate. We conjecture that proving optimal regret bounds for any algorithm requires tight concentration results for the number of times a bandit algorithm pulls a suboptimal arm within a regenerative cycle whose length is dependent on the bandit strategy.

Although we analyze only a special case of the model, i.e., when there is unique optimal matching between the queues and the servers, this opens up exciting new directions for future research. We discuss some of them below:

1. **General Case:**

   An obvious direction of research is to generalize these results to the case where no matching is optimal for *all* users. In this case, two challenges arise:

   (i) **Characterization of the Optimal Policy**: In the absence of a single matching that is optimal for all users, the *optimal* policy is a subject of debate. Since the capacity region can be characterized by the class of all static-split rules, one could pick a static-split policy that achieves some point on the pareto boundary and compare the queue performance with respect to this policy. It is necessary to explore the right choice of static-split policy for a given arrival rate vector.

   (ii) **Challenges in Analysis**: With progression from the unique matching

case to the general case, the problem has been transformed from that of hypothesis testing to an estimation problem. The parameters to be estimated in this case are the factors that determine the convex combination of the static split rule. While a simple structured exploration component was sufficient in accurately identifying the optimal matching in Chapter 3, it is not clear if the same strategy would work for the estimation problem.

2. **Partial Channel State Information:**

   The main motivation of formulating the scheduling problem as a multi-armed bandit problem is the prohibitive cost involved in obtaining channel state information for all the user-server link pairs. However, it might be feasible to obtain the channel states for a restricted number of links. This leads us to pose the following extension of the scheduling problem. Suppose that in each time-slot, it is possible to obtain CSI for at most m matchings in the bipartite graph before the scheduling decision is made and exactly one of these m matchings can be scheduled after the observation. Gopalan *et al.* characterize the capacity region under this setting and propose an queue-length based algorithm that achieves the optimal exponential rate decay in steady state [61]. It would be interesting to explore bandit-style algorithms in this setting and obtain finite time regret bounds.

3. **Dependent Links:**

   Another evident setting of interest is one where the service of links are statistically dependent. This could be due to some underlying causal graph or an

inherently smaller dimension of the problem.

5.1.3 Scheduling with Energy Costs

We note that the solution proposed requires the scheduler to solve a potentially large LP in every time-slot. It would be useful to explore direct methods of adaptive control for this problem – a scheduling algorithm that directly computes the activation and rate allocation policy without computing the underlying statistical parameters. For example, is it possible to design a scheduling algorithm using principles of reinforcement learning? An immediate difficulty is the continuum of the state space which includes the queue-length vector. Another potential difficulty is the lack of access to instantaneous channel rates at the instant of BS activation[1].

The basic model considered in this thesis can be extended to include many other features of cellular networks – user mobility, network densification etc. It would be interesting to explore learning-cum-allocation algorithms for these more complex models.

[1]Recall that the instantaneous channel rates can be observed only after the BSs have been activated.

Appendices

Appendix A

Error Analysis for *BiAD*

A.1 Proof of Theorem 2.1

We first state and prove two lemmas used in proving the main theorem (Theorem 2.1).

Lemma A.1. *For any* objective *recommendation algorithm and for any user u, item i, and time $t < (F(\hat{R}_{ui}(t), \hat{\mathbf{R}}_u(t)) + 1)$, the probability that item i is recommended to user u at time t is upper bounded by*

$$W_{ui}(t) \leq \frac{1}{F\left(\hat{R}_{ui}(t), \hat{\mathbf{R}}_u(t)\right) - t + 1} \,. \tag{A.1}$$

Proof.

$$
\begin{aligned}
1 &\overset{(a)}{=} \sum_{j=1}^{m} W_{uj}(t) \\
&\geq \sum_{j=1}^{m} \mathbb{1}\left\{\hat{R}_{uj}(t) \geq \hat{R}_{ui}(t)\right\} \left(1 - \sum_{l=1}^{t-1} \mathbb{1}_{ui}(l)\right) W_{uj}(t) \\
&\overset{(b)}{\geq} \sum_{j=1}^{m} \mathbb{1}\left\{\hat{R}_{uj}(t) \geq \hat{R}_{ui}(t)\right\} \left(1 - \sum_{l=1}^{t-1} \mathbb{1}_{ui}(l)\right) W_{ui}(t) \\
&= \left|\left\{j : \hat{R}_{uj}(t) \geq \hat{R}_{ui}(t), \sum_{l=1}^{t-1} \mathbb{1}_{uj}(l) = 0\right\}\right| W_{ui}(t) \\
&\geq \left(\left|\left\{j : \hat{R}_{uj}(t) \geq \hat{R}_{ui}(t)\right\}\right| - \left|\left\{j : \sum_{l=1}^{t-1} \mathbb{1}_{uj}(l) \neq 0\right\}\right|\right) W_{ui}(t) \\
&= \left(F\left(\hat{R}_{ui}(t), \hat{\mathbf{R}}_u(t)\right) - (t-1)\right) W_{ui}(t),
\end{aligned}
$$

where the (a) and (b) follow from the characterization of an *objective* recommendation algorithm in Section 2.2.2.1 – equality (a) due to the fact that $\mathbf{W}(t)$ is a stochastic matrix (Property 1), and inequality (b) due to the monotonic property satisfied by the weight matrix (Property 3). The above inequality gives the desired bound in (A.1) for $t < (F(\hat{R}_{ui}(t), \hat{\mathbf{R}}_u(t)) + 1)$.

Lemma A.2. *Let* $\{X_i,\, i = 1, \ldots, k\}$ *be independent random variables with range* $[0, 1]$ *and mean* $\{p_i,\, i = 1, \ldots, k\}$*, and let* $\sum_{i=1}^{k} p_i \leq p < k$*. Then,*

$$
\mathbb{P}\left[\sum_{i=1}^{k} X_i \geq T\right] \leq \exp\left(-T \log\left(\frac{T}{p}\right) + T - p\right) \forall\, k > T > p.
$$

Using Chernoff bound for independent random variables, we have, for any

$\theta > 0$,

$$\begin{aligned}
\mathbb{P}\left[\sum_{i=1}^{k} X_i \geq T\right] &\leq e^{-\theta T} \prod_{i=1}^{k} \mathbb{E}\left[e^{\theta X_i}\right] \\
&= e^{-\theta T} \prod_{i=1}^{k} \left(p_i e^{\theta} + 1 - p_i\right) \\
&\leq e^{-\theta T} \left(\frac{1}{k} \sum_{i=1}^{k} \left(p_i e^{\theta} + 1 - p_i\right)\right)^k \\
&\leq e^{-\theta T} \left(\frac{p}{k}\left(e^{\theta} - 1\right) + 1\right)^k,
\end{aligned}$$

where the second inequality follows from the fact that the geometric mean of non-negative numbers is at most their arithmetic mean, and the last inequality follows for $\theta > 0$, which is true for the choice of $\theta = \log\left((k-p)T/(p(k-T))\right)$ for any $T > p$. Then, we get

$$\begin{aligned}
\mathbb{P}\left[\sum_{i=1}^{k} X_i \geq T\right] &\leq \left(\frac{p(k-T)}{(k-p)T}\right)^T \left(\frac{(k-p)T}{(k-T)k} + 1 - \frac{p}{k}\right)^k \\
&= \left(\frac{p}{T}\right)^T \left(\frac{k-p}{k-T}\right)^{k-T} \\
&= \left(\frac{p}{T}\right)^T \left(1 + \frac{T-p}{k-T}\right)^{k-T} \\
&\leq \left(\frac{p}{T}\right)^T e^{T-p} \\
&= \exp\left(-T \log\left(\frac{T}{p}\right) + T - p\right).
\end{aligned}$$

□

Proof of Theorem 2.1. The proof of the theorem consists of two parts which give upper bounds for probability of Type I and Type II errors. For any user u and round l, let $\Xi_u(l)$ be the Bernoulli random variable with mean ξ_u indicating if the feedback given by user u in round l is in error.

Type I Error

The algorithm makes a Type I error if it declares an *objective* recommendation engine to be *biased*. This section shows that the probability that the algorithm makes Type I Error is low.

Suppose that the recommendation engine uses an *objective* recommendation algorithm. We first bound the probability that $BiAD$ accepts H_1 in round t. Recall that the algorithm accepts H_1 in round t if $S(t) \geq T(t)$, and that

$$S(t) = \max_{\left\{\hat{\mathcal{A}} \subseteq [m] : |\hat{\mathcal{A}}| = \hat{A}(t)\right\}} \sum_{i \in \hat{\mathcal{A}}} B_i(t).$$

Consider a fixed $\hat{\mathcal{A}} \subseteq [m]$ such that $|\hat{\mathcal{A}}| = \hat{A}(t)$. We first bound the probability that $\sum_{i \in \hat{\mathcal{A}}} B_i(t) \geq T(t)$ and then use union bound over all possible $\hat{\mathcal{A}}$ to obtain an upper bound on the probability that $S(t) \geq T(t)$. Recall that $\Xi_u(l)$ is equal to 1 if the feedback given by user u in round l is in error and 0 otherwise. Let us define

$$X_u(l) := \sum_{i \in \hat{\mathcal{A}}} \mathbb{1}_{ui}(l) \left(\mathbb{1}\left\{R_{ui} \geq \eta\right\} \Xi_u(l) + \mathbb{1}\left\{R_{ui} < \eta\right\} (1 - \Xi_u(l)) \right). \tag{A.2}$$

Note that $X_u(l)$ is equal to 1 if, in round l, an item from set $\hat{\mathcal{A}}$ is recommended to player u and the player declares it ineffective. It is equal to 0 otherwise. Given $\left\{\mathbf{W}(l), \hat{\mathbf{R}}(l),\, l \in [t]\right\}$, we have that $\{X_u(l),\, l \in [t],\, u \in [n]\}$ are Bernoulli random variables independent of each other. For every $1 \leq u \leq n$ and $1 \leq l \leq t$, the mean of $X_u(l)$ can be bounded as follows:

$$\mathbb{E}\left[X_u(l)\middle|\left\{\mathbf{W}(l'),\hat{\mathbf{R}}(l')\right\}_{l'=1}^{t}\right] \leq \sum_{i\in\hat{\mathcal{A}}} W_{ui}(l)\left(\xi_u + \mathbb{1}\left\{R_{ui}<\eta\right\}\right)(1-\xi_u)$$
$$\leq \xi_u + P_u^{\hat{\mathcal{A}}}(l)\,(1-\xi_u)\,, \tag{A.3}$$

where the inequality follows from the upper bound for $W_{ui}(l)$ from Lemma A.1 and the definition of $P_u^{\hat{\mathcal{A}}}(l)$ ((2.7)). From (2.6) and Condition (I)a, we have

$$\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u + \mathbb{E}\left[P_u^{\hat{\mathcal{A}}}(l)\right](1-\xi_u) \leq p(t).$$

Note that, since the noise in the rating estimates are independent across time and users, $\left\{P_u^{\hat{\mathcal{A}}}(l),\, l\in[t],\, u\in[n]\right\}$ are independent random variables. Therefore, we can use the Chernoff bound in Lemma A.2 to obtain a probabilistic upper bound on their sum. By Condition (I)b, $p(t) < nt$. In addition, to prove that $\hat{p}(t) < nt$, we consider the following two cases:

(i): $\beta(t) \geq 1$

$$\begin{aligned}
\hat{p}(t) &= \exp\left(1 + W\left(\frac{\beta(t)}{e}\right)\right)p(t)\\
&= \frac{\beta(t)p(t)}{W\left(\frac{\beta(t)}{e}\right)}\\
&\leq \frac{\left(\hat{A}(t)+c\right)\log m}{W\left(\frac{\beta(t)}{e}\right)}\\
&\leq 6\hat{A}(t)\log m\\
&< nt
\end{aligned}$$

for n large enough by Condition (I)c. The second inequality follows from the fact that $\beta(t) \geq 1$ implies $\left(6W\left(\frac{\beta(t)}{e}\right) - 1\right)\hat{A}(t) \geq 0.5 = c$.

(ii): $\beta(t) < 1$

$$\begin{aligned}\hat{p}(t) &= \exp\left(1 + W\left(\frac{\beta(t)}{e}\right)\right) p(t) \\ &< 4p(t) \\ &\leq nt\end{aligned}$$

where, the first inequality follows from the fact that $\beta(t) < 1$ implies $\exp\left(1 + W\left(\frac{\beta(t)}{e}\right)\right) < 4$ and the second inequality follows from Condition (I)b.

Applying Lemma A.2 gives

$$\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u + P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right) \geq \hat{p}(t)\right] \leq \exp\left(-\hat{p}(t)\left(\log\left(\frac{\hat{p}(t)}{p(t)}\right) - 1\right) - p(t)\right).$$

Now, by the definition of Lambert-W function,

$$\hat{p}(t)\left(\log\left(\frac{\hat{p}(t)}{p(t)}\right) - 1\right) \geq \left(\hat{A}(t) + c\right)\log m - p(t),$$

which further implies that

$$\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u + P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right) \geq \hat{p}(t)\right] \leq \exp\left(-\left(\hat{A}(t) + c\right)\log m\right). \tag{A.4}$$

We now proceed to obtain a probabilistic upper bound for the sum, $\sum_{u=1}^{n}\sum_{l=1}^{t} X_u(l)$. By inequality (A.3), the sum of the corresponding means has the following upper bound:

$$\sum_{u=1}^{n}\sum_{l=1}^{t}\mathbb{E}\left[X_u(l)\middle|\left\{\mathbf{W}(l'),\hat{\mathbf{R}}(l')\right\}_{l'=1}^{t}\right]\leq\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u+P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right).$$

Since $\{X_u(l),\, l\in[t],\, u\in[n]\}$ are independent Bernoulli random variables given $\left\{\mathbf{W}(l),\hat{\mathbf{R}}(l),\, l\in[t]\right\}$, Lemma A.2 can be used as before. If $\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u+P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right)\leq\hat{p}(t)$, then Lemma A.2 gives

$$\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}X_u(l)\geq T(t)\middle|\left\{\mathbf{W}(l'),\hat{\mathbf{R}}(l')\right\}_{l'=1}^{t}\right]\leq\exp\left(-\left(\hat{A}(t)+c\right)\log m\right).\quad\text{(A.5)}$$

The above upper bound can be derived in exactly the same manner as the upper bound in (A.4). Combining this upper bound with inequality (A.4), we have

$$\begin{aligned}\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}X_u(l)\geq T(t)\right]&\leq\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}X_u(l)\geq T(t)\middle|\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u+P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right)\leq\hat{p}(t)\right]\\&\quad+\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{t}\xi_u+P_u^{\hat{\mathcal{A}}}(l)\left(1-\xi_u\right)\geq\hat{p}(t)\right]\\&\leq 2\exp\left(-\left(\hat{A}(t)+c\right)\log m\right).\end{aligned}\quad\text{(A.6)}$$

Now, recall that $B_i(t)$ is the number of players who have rated item i ineffective upto round t, which can be mathematically written as $B_i(t)=\sum_{u=1}^{n}\sum_{l=1}^{t}\mathbb{1}_{ui}(l)\cdot\mathbb{1}\left\{R_{ui}<\eta\right\}$. Using definition (A.2), we have

$$\sum_{i\in\hat{\mathcal{A}}}B_i(t)=\sum_{u=1}^{n}\sum_{l=1}^{t}X_u(l),$$

which gives us the following equivalent form of inequality (A.6):

$$\mathbb{P}\left[\sum_{i\in\hat{\mathcal{A}}}B_i(t)\geq T(t)\right]\leq 2\exp\left(-\left(\hat{A}(t)+c\right)\log m\right).\quad\text{(A.7)}$$

We can now take a union bound over all possible $\hat{\mathcal{A}}$ to bound the probability that *BiAD* accepts H_1 in round t.

$$\begin{aligned}
\mathbb{P}\left[S(t) \geq T(t)\right] &= \mathbb{P}\left[\bigcup_{\left\{\hat{\mathcal{A}}\subseteq[m]:|\hat{\mathcal{A}}|=\hat{A}(t)\right\}} \left\{\sum_{i\in\hat{\mathcal{A}}} B_i(t) \geq T(t)\right\}\right] \\
&\leq \sum_{\left\{\hat{\mathcal{A}}\subseteq[m]:|\hat{\mathcal{A}}|=\hat{A}(t)\right\}} \mathbb{P}\left[\sum_{i\in\hat{\mathcal{A}}} B_i(t) \geq T(t)\right] \\
&\leq 2\binom{m}{\hat{A}(t)} \exp\left(-\left(\hat{A}(t)+c\right)\log m\right) \\
&\leq 2\exp\left(-c\log m\right) \\
&= 2m^{-c},
\end{aligned}$$

where the second inequality follows from (A.7). Further taking a union bound over all rounds,

$$\begin{aligned}
\mathbb{P}[\text{Type } I \text{ Error}] &= \mathbb{P}\left[\cup_{t=1}^{Q(m)} S(t) \geq T(t)\right] \\
&\leq \sum_{t=1}^{Q(m)} \mathbb{P}\left[S(t) \geq T(t)\right] \\
&\leq 2\,Q(m)\,m^{-c}.
\end{aligned}$$

This shows that *BiAD* declares an *objective* recommendation engine as *biased* with probability $O(\frac{Q(m)}{\sqrt{m}})$ for the choice of $c = 1/2$.

Type II Error

The algorithm makes a Type II error if it does not detect a *biased* recommendation engine, i.e., it declares H_0 when H_1 is true. Suppose that the recommendation engine is *biased* with an ad-pool $\mathcal{A}$ of size $|\mathcal{A}| = A$. By Condition (II)a, BiAD

has at least A rounds of feedback. We prove that the total number of ad recommendations declared ineffective by the n players until round A is at least $\gamma\Omega(nA)$ with high probability. Let $\delta \in (0,1)$ be the constant given by Condition (II)b. For any $1 \leq u \leq n$, let $Y_u(l) = 1$ if the *biased* recommendation engine decides to recommend from the ad-pool to user u in round l. Since $\{(1 - \Xi_u(l))\, Y_u(l),\, l \in [A],\, u \in [n]\}$ are independent Bernoulli random variables with mean $\{(1 - \xi_u)\, \gamma,\, u \in [n]\}$, we have

$$\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{A}(1 - \Xi_u(l))\, Y_u(l) < (1 - \delta/2)(1 - c')\gamma An\right] \leq e^{-\frac{\delta^2}{8}(1-c')\gamma An},$$

where the inequality follows from a version of Chernoff bound given in [105] for sum of i.i.d. Bernoulli random variables using Condition (I)a. Note that

$$\sum_{u=1}^{n}\sum_{l=1}^{A}(1 - \Xi_u(l))\sum_{i\in\mathcal{A}}\mathbb{1}_{ui}(l)$$

is the total number of recommendations from the ad-pool $\mathcal{A}$ that were rated without error by the players and also that $\sum_{i\in\mathcal{A}}\mathbb{1}_{ui}(l) \geq Y_u(l)$. Therefore,

$$\mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{A}(1 - \Xi_u(l))\sum_{i\in\mathcal{A}}\mathbb{1}_{ui}(l) < (1 - \delta/2)(1 - c')\gamma nA\right] \leq e^{-\frac{\delta^2}{8}(1-c')\gamma nA}. \quad \text{(A.8)}$$

The detection algorithm makes the correct decision if in round t, $S(t) \geq T(t)$ for some $t \leq Q(m)$. We show that $S(A) \geq T(A)$ with high probability. Since $A \leq Q(m)$, the algorithm makes the correct decision with high probability. Now, suppose that the total number of ad recommendations rated without error by the players until round A is at least $(1 - \delta/2)(1 - c')\gamma nA$. Since, by Condition (II)c, the total number of effective ads to the n players is $o(\gamma nA)$, the total number of recommendations from the ad-pool rated ineffective until round A is $\gamma\Omega(nA)$.

Consequently,

$$\begin{aligned} S(A) &= \max_{\{\hat{\mathcal{A}}\subseteq[m]:|\hat{\mathcal{A}}|=A\}} \sum_{i\in\hat{\mathcal{A}}} B_i(t) \\ &\geq \sum_{i\in\mathcal{A}} B_i(t) \\ &\geq (1-\delta/2)(1-c')\gamma nA - o(\gamma nA) \\ &\geq (1-\delta)(1-c')\gamma nA, \end{aligned} \tag{A.9}$$

for n large enough. To prove that $T(A)$ does not exceed the right hand side of inequality (A.9), we consider the following cases:

(i): $\hat{\beta}(A) \geq e$

Since $W(\cdot)$ is an increasing function in $[0,\infty)$, we have $W\left(\frac{\hat{\beta}(A)}{e}\right) \geq W(1) > \frac{1}{2}$. Now,

$$\begin{aligned} T(A) &= \exp\left(1 + W\left(\frac{\hat{\beta}(A)}{e}\right)\right)\hat{p}(A) \\ &= \frac{\hat{\beta}(A)}{W\left(\frac{\hat{\beta}(A)}{e}\right)}\hat{p}(A) \\ &\leq \frac{(A+c)\log m}{W(1)} \\ &= \frac{(A+c)}{W(1)A}\frac{\log m}{n}(nA) \\ &\leq \left(\frac{e}{W(1)}\right)^2 \frac{\log m}{n}(nA). \end{aligned}$$

where the second equality follows from the definition of the Lambert W function, and the first inequality follows by using the definition of $\hat{\beta}(A)$ given by (2.3).

(ii): $\hat{\beta}(A) < e,\ \beta(A) \geq e$

$$\begin{aligned}
\hat{p}(A) &= \exp\left(1 + W\left(\frac{\beta(A)}{e}\right)\right) p(A) \\
&= \frac{\beta(A)}{W\left(\frac{\beta(A)}{e}\right)} p(A) \\
&\leq \frac{(A+c)\log m}{W(1)}. \\
T(A) &= \exp\left(1 + W\left(\frac{\hat{\beta}(A)}{e}\right)\right) \hat{p}(A) \\
&\leq \exp\left(1 + W(1)\right) \hat{p}(A) \\
&= \frac{\exp\left(1 + W(1)\right)}{W(1)} \frac{(A+c)}{A} \frac{\log m}{n} (nA) \\
&\leq \left(\frac{e}{W(1)}\right)^2 \frac{\log m}{n} (nA).
\end{aligned}$$

(iii): $\hat{\beta}(A) < e,\ \beta(A) < e$

$$\begin{aligned}
T(A) &= \exp\left(1 + W\left(\frac{\hat{\beta}(A)}{e}\right)\right) \hat{p}(A) \\
&\leq \exp\left(1 + W(1)\right) \hat{p}(A) \\
&\leq \exp\left(2 + 2W(1)\right) p(A) \\
&= \left(\frac{e}{W(1)}\right)^2 \frac{p(A)}{nA} nA.
\end{aligned}$$

Combining the results from the above three cases with inequality (A.9) and Condition (II)b gives that $S(A) \geq T(A)$ for n large enough. Therefore, the algorithm declares the correct hypothesis in round A if the total number of ad recommendations rated without error by the players until round A is at least $(1-\delta/2)(1-c')\gamma nA$. We can therefore use the concentration inequality in (A.8) to bound the probability

of Type II error.

$$\begin{aligned}
\mathbb{P}[\text{Type } II \text{ Error}] &\leq \mathbb{P}\left[S(A) < T(A)\right] \\
&\leq \mathbb{P}\left[\sum_{u=1}^{n}\sum_{l=1}^{A}(1-\Xi_u(l))\sum_{i\in\mathcal{A}}\mathbb{1}_{ui}(l) < (1-\delta/2)(1-c')\gamma nA\right] \\
&\leq e^{-\frac{\delta^2}{8}(1-c')\gamma nA} \\
&= e^{-\Omega((1-c')\gamma n)}.
\end{aligned}$$

This shows that the probability of Type II error decays exponentially with the number of players and the bias probability. □

Appendix B

Queue-Regret Analysis for Queueing Bandits

B.1 Q-ThS Algorithm

Figure 4 shows the scheduling algorithm Q-ThS, which is a structured-explore variant of Thompson Sampling. All results presented in Chapter 3 for Q-UCB also hold for Q-ThS.

B.2 Proofs

We provide details of the proofs for Theorem 3.2 in Section B.2.1 and for Theorems 3.1 and 3.3 in Section B.2.2. In each section, we state and prove a few intermediate lemmas that are useful in proving the theorems. All the theorems and lemmas that hold for Q-UCB also hold for Q-ThS.

B.2.1 Regret Upper Bound for Q-UCB

As shown in Algorithm 2, $\mathsf{E}(t)$ indicates whether Q-UCB chooses to explore at time t. We now obtain a bound on the expected number of time-slots Q-UCB chooses to explore in an arbitrary time interval $(t_1, t_2]$. Since at any time t, Q-UCB decides to explore with probability $\min\{1, 3K\frac{\log^2 t}{t}\}$, we have

$$\mathbb{E}\left[\sum_{l=t_1+1}^{t_2} \mathsf{E}(l)\right] \leq 3K \sum_{l=t_1+1}^{t_2} \frac{\log^2 l}{l} \leq 3K \int_{t_1}^{t_2} \frac{\log^2 l}{l}\, \mathsf{d}l = K\left(\log^3 t_2 - \log^3 t_1\right). \tag{B.1}$$

Algorithm 4 Q-ThS

At time t,

Let $\mathsf{E}(t)$ be an independent Bernoulli sample of mean $\min\{1, 3K\frac{\log^2 t}{t}\}$.

if $\mathsf{E}(t) = 1$ **then**

Explore:

Schedule a matching from $\mathcal{E}$ uniformly at random.

else

Exploit:

For each $k \in [K], u \in [U]$, pick a sample $\hat{\theta}_{uk}(t)$ of distribution,

$$\hat{\theta}_{uk}(t) \sim \text{Beta}\left(\hat{\mu}_{uk}(t)T_{uk}(t) + 1, \left(1 - \hat{\mu}_{uk}(t)\right)T_{uk}(t) + 1\right).$$

Compute for all $u \in [U]$

$$\hat{k}_u(t) := \arg\max_{k\in[K]} \hat{\theta}_{uk}(t)$$

Schedule a matching $\boldsymbol{\kappa}(t)$ such that

$$\boldsymbol{\kappa}(t) \in \arg\min_{\boldsymbol{\kappa}\in\mathcal{M}} \sum_{u\in[U]} \mathbb{1}\left\{\kappa_u \neq \hat{k}_u(t)\right\},$$

i.e., $\boldsymbol{\kappa}(t)$ *is the projection of* $\hat{\boldsymbol{k}}(t)$ *onto the space of all matchings* $\mathcal{M}$ *with Hamming distance as metric.*

end if

The following lemma gives a probabilistic upper bound on the same quantity.

Lemma B.1. *For any t and $t_1 < t_2$,*

$$\mathbb{P}\left[\sum_{l=t_1+1}^{t_2} \mathsf{E}(l) \geq 5\max\left(\log t, K\left(\log^3 t_2 - \log^3 t_1\right)\right)\right] \leq \frac{1}{t^4}.$$

Proof. To prove the result, we will use the following Chernoff bound: for a sum of independent Bernoulli random variables Y with mean $\mathbb{E}Y$ and for any $\delta > 0$,

$$\mathbb{P}\left[[Y \geq (1+\delta)\mathbb{E}Y\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\mathbb{E}Y}.$$

If $\mathbb{E}Y \geq \log t$, the above bound for $\delta = 4$ gives

$$\mathbb{P}\left[Y \geq 5\mathbb{E}Y\right] \leq \frac{1}{t^4}.$$

Note that $\{\mathsf{E}(l)\}_{l=t_1+1}^{t_2}$ are independent Bernoulli random variables and let $X = \sum_{l=t_1}^{t_2} \mathsf{E}(l)$. Now consider the probability $\mathbb{P}\left[X \geq 5\max\left(\log t, \mathbb{E}X\right)\right]$. If $\mathbb{E}X \geq \log t$, then the result is true from the above Chernoff bound. If $\mathbb{E}X < \log t$, then it is possible to construct a random variable Y which is a sum of independent Bernoulli random variables, has mean $\log t$ and stochastically dominates X, in which case we can again use the Chernoff bound on Y. Therefore,

$$\mathbb{P}\left[X \geq 5\log t\right] \leq \mathbb{P}\left[Y \geq 5\log t\right] \leq \frac{1}{t^4}.$$

Using inequality (B.1), we have the required result, i.e.,

$$\mathbb{P}\left[\sum_{l=t_1+1}^{t_2} \mathsf{E}(l) \geq 5\max\left(\log t, K\left(\log^3 t_2 - \log^3 t_1\right)\right)\right] \leq \mathbb{P}\left[X \geq 5\max\left(\log t, \mathbb{E}X\right)\right] \leq 1/t^4.$$

□

Let $w(t) = \exp\left(\left(\frac{2\log t}{\Delta}\right)^{2/3}\right)$. The next lemma shows that, with high probability, Q-UCB (or Q-ThS) does not schedule a sub-optimal matching when it exploits in the late stage.

Lemma B.2.

$$\mathbb{P}\left[\bigcup_{u\in[U]}\sum_{l=w(t)+1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) > 0\right] = O\left(\frac{UK}{t^3}\right).$$

Proof. Let $X_{uk}(l), u = 1, 2, \ldots, K, k = 1, 2, \ldots, K, l = 1, 2, 3, \ldots$ be independent random variables denoting the service offered in the l^{th} assignment of server k to queue u and let $B_{uk}(s,t) = \frac{1}{s}\sum_{l=1}^{s} X_{uk}(l) + \sqrt{\frac{\log^2 t}{2s}}$. Consider the events

$$T_{uk}(l) \geq \frac{1}{2}\log^3(l-1) \quad \forall k \in [K], u \in [U], w(t)+1 \leq l \leq t, \tag{B.2}$$

$$B_{uk_u^*}(s,l) > \mu_u^* \quad \forall s,l \text{ s.t. } \frac{1}{2}\log^3(w(t)) \leq s \leq t-1, w(t)+1 \leq l \leq t, \ \forall u \in [U], \tag{B.3}$$

and

$$B_{uk}(s,l) \leq \mu_u^* \quad \forall s,l \text{ s.t. } \frac{1}{2}\log^3(w(t)) \leq s \leq t-1, w(t)+1 \leq l \leq t, \forall k \neq k_u^*, \ \forall u \in [U]. \tag{B.4}$$

It can be seen that, given the above events, Q-UCB schedules the optimal matching in all time-slots in $(w(t), t]$ in which it decides to exploit, i.e., $\sum_{l=w(t)+1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) = 0$ for all $u \in [U]$. We now show that the events above occur with high probability.

Note that, since the matchings in $\mathcal{E}$ cover all the links in the system, $T_{uk}(l+1) \leq \frac{1}{2}\log^3(l)$ for some u, k implies that $\sum_{s=1}^{l}\mathbb{1}\{\boldsymbol{\kappa}(s) = \boldsymbol{\kappa}\} \leq \frac{1}{2}\log^3(l)$ for some $\boldsymbol{\kappa} \in \mathcal{E}$. Since $\sum_{s=1}^{l}\mathbb{1}\{\boldsymbol{\kappa}(s) = \boldsymbol{\kappa}\}$ is a sum of i.i.d. Bernoulli random variables with

sum mean at least $\log^3(l)$, we use Chernoff bound to prove that event (B.2) occurs with high probability.

$$\begin{aligned}
\mathbb{P}\left[\text{(B.2) is false}\right] &\leq \sum_{\boldsymbol{\kappa}\in\mathcal{E}} \sum_{l=w(t)}^{t-1} \mathbb{P}\left[\sum_{s=1}^{l} \mathbb{1}\left\{\boldsymbol{\kappa}(s)=\boldsymbol{\kappa}\right\} \leq \frac{1}{2}\log^3(l)\right] \\
&\leq Kt\exp\left(-\frac{1}{8}\log^3(w(t))\right) \\
&= Kt\exp\left(-\frac{1}{8}\left(\frac{2\log t}{\Delta}\right)^2\right) = o\left(\frac{K}{t^4}\right). \qquad \text{(B.5)}
\end{aligned}$$

Similarly, probability of events (B.3) and (B.4) can be bounded as follows –

$$\begin{aligned}
\mathbb{P}\left[\text{(B.3) is false}\right] &\leq \sum_{u\in[U]} \sum_{l=w(t)+1}^{t} \sum_{s=\frac{1}{2}\log^3(w(t))}^{t-1} \mathbb{P}\left[B_{uk_u^*}(s,l) \leq \mu_u^*\right] \\
&\leq U \sum_{l=w(t)+1}^{t} \sum_{s=\frac{1}{2}\log^3(w(t))}^{t-1} \exp\left(-\log^2(l)\right) \\
&\leq U\exp\left(-\log^2(w(t)) + 2\log t\right) \\
&= U\exp\left(-\left(\frac{2\log t}{\Delta}\right)^{4/3} + 2\log t\right) = o\left(\frac{U}{t^4}\right). \qquad \text{(B.6)}
\end{aligned}$$

$$\begin{aligned}
\mathbb{P}\left[\text{(B.4) is false}\right] &\leq \sum_{u\in[U],k\neq k_u^*} \sum_{l=w(t)+1}^{t} \sum_{s=\frac{1}{2}\log^3(w(t))}^{t-1} \mathbb{P}\left[B_{uk}(s,l) > \mu_u^*\right] \\
&\leq \sum_{u\in[U],k\neq k_u^*} \sum_{l=w(t)+1}^{t} \sum_{s=\frac{1}{2}\log^3(w(t))}^{t-1} \mathbb{P}\left[B_{uk}(s,l) > \Delta + \mu_{uk}\right] \\
&\leq UK \sum_{l=w(t)+1}^{t} \sum_{s=\frac{1}{2}\log^3(w(t))}^{t-1} \exp\left(-2s\left(\Delta - \sqrt{\frac{\log^2 l}{2s}}\right)^2\right) \\
&\leq UKt^2\exp\left(-\log^3(w(t))\left(\Delta - \sqrt{\frac{\log^2 t}{\log^3(w(t))}}\right)^2\right) \\
&= UK\exp\left(-\log^2 t + 2\log t\right) = o\left(\frac{UK}{t^3}\right). \qquad \text{(B.7)}
\end{aligned}$$

Combining the inequalities (B.5), (B.6) and (B.7) we have

$$\mathbb{P}\left[\bigcup_{u\in[U]}\sum_{l=w(t)+1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l)>0\right] \leq \mathbb{P}\left[\text{(B.2) is false}\right]+\mathbb{P}\left[\text{(B.3) is false}\right]+\mathbb{P}\left[\text{(B.4) is false}\right]$$
$$= O\left(\frac{UK}{t^3}\right).$$

This proves the result for Q-UCB.

The proof of this result for Q-ThS follows in a similar fashion. For Q-ThS, events (B.3) and (B.4) are substituted with the following events

$$\theta_{uk_u^*}(s) > \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}},\ \forall s,\ \text{s.t. } w(t)+1\leq s\leq t, u\in[U] \tag{B.8}$$

and

$$\theta_{uk}(s) \leq \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}},\ \forall s,k \text{ s.t. } w(t)+1\leq s\leq t, k\neq k_u^*, u\in[U]. \tag{B.9}$$

It is then sufficient to prove that the above two events occur with high probability. Given events (B.2), (B.8), (B.9), Q-ThS schedules the optimal matching in all time-slots in $(w(t),t]$ in which it decides to exploit, i.e., $\sum_{l=w(t)+1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l)=0$ for all $u\in[U]$.

Let $\Sigma_{u,k,l}=\sum_{r=1}^{l}X_{uk}(r)$ and $S_{uk}(l)=\hat{\mu}_{uk}(l)T_{uk}(l)=\Sigma_{u,k,T_{uk}(l)}$ for all $u\in[U], k\in[K]$, $l\in\mathbb{N}$. We use the 'Beta-Binomial trick' (used in [76, 13]), which gives a relation between the c.d.fs of Beta and Binomial distributions to prove the high probability results. Let $F_{a,b}^{\text{Beta}}$ and $F_{n,p}^{\text{B}}$ denote the c.d.f of Beta(a,b) distribution and the c.d.f. of Binomial(n,p) distribution respectively. Then

$$F_{a,b}^{\text{Beta}}(y) = 1 - F_{a+b-1,y}^{\text{B}}(a-1).$$

For each $s, l \in \mathbb{N}$, let $\{Z_{s,l}(r)\}_{r>0}$ be a sequence of i.i.d. Bernoulli random variables with mean $\mu_u^* - \frac{\log(s-1)}{\sqrt{2l}}$. Now, to bound probability of event (B.8),

$$
\begin{aligned}
&\mathbb{P}\left[\text{(B.8) is false}\right] \\
&\leq \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \mathbb{P}\left[\theta_{uk_u^*}(s) \leq \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}}\right] \\
&= \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \mathbb{E}\left[F^{\text{Beta}}_{S_{uk_u^*}(s)+1, T_{uk_u^*}(s)-S_{uk_u^*}(s)+1}\left(\mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}}\right)\right] \\
&= \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \mathbb{E}\left[1 - F^{\text{B}}_{T_{uk_u^*}(s)+1, \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}}}\left(S_{uk_u^*}(s)\right)\right] \\
&= \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \sum_{l=0}^{s-1} \mathbb{P}\left[T_{uk_u^*}(s) = l\right] \mathbb{E}\left[1 - F^{\text{B}}_{l+1, \mu_u^* - \frac{\log(s-1)}{\sqrt{2l}}}\left(\Sigma_{u,k_u^*,l}\right) \,\middle|\, T_{uk_u^*}(s) = l\right] \\
&\leq \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \left(\mathbb{P}\left[T_{uk_u^*}(s) < \frac{1}{2}\log^3(s-1)\right] + \sum_{l=\frac{1}{2}\log^3(s-1)}^{s} \mathbb{P}\left[\Sigma_{u,k_u^*,l} \leq \sum_{r=1}^{l+1} Z_{s,l}(r)\right]\right) \\
&\leq o\left(\frac{UK}{t^3}\right) + \sum_{u\in[U]} \sum_{s=w(t)+1}^{t} \sum_{l=\frac{1}{2}\log^3(s-1)}^{s} \exp\left(-\frac{\log^2(s-1)}{6}\right) = o\left(\frac{UK}{t^3}\right).
\end{aligned}
$$

The last inequality follows by using (B.5) to bound the first term and Chernoff-Hoeffding inequality to bound the second term.

Similarly, the probability of event (B.9) can be bounded as follows.

$$
\begin{aligned}
&\mathbb{P}\left[\text{(B.9) is false}\right] \\
&\leq \mathbb{P}\left[\text{(B.2) is false}\right] + \sum_{u\in[U], k\neq k_u^*} \sum_{s=w(t)+1}^{t} \mathbb{P}\left[\theta_{uk}(s) > \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}} \bigcap \text{(B.2) is true}\right]
\end{aligned}
$$

Now, for any $u \in [U], k \neq k_u^*$, $w(t)+1 \leq s \leq t$, again using the 'Beta-Binomial

trick', we have

$$\begin{aligned}
&\mathbb{P}\left[\theta_{uk}(s) > \mu_u^* - \frac{\log(s-1)}{\sqrt{2T_{uk_u^*}(s)}} \bigcap \text{(B.2) is true}\right] \\
&\leq \sum_{l=\frac{1}{2}\log^3(s-1)}^{s} \mathbb{P}\left[T_{uk_u^*}(s) = l\right] \mathbb{E}\left[F^{\mathrm{B}}_{l+1,\mu_u^* - \frac{\log(s-1)}{\sqrt{2l}}}\left(\Sigma_{u,k,l}\right) \,\middle|\, T_{uk_u^*}(s) = l\right] \\
&\leq \sum_{l=\frac{1}{2}\log^3(s-1)}^{s} \mathbb{P}\left[\sum_{r=1}^{l+1} Z_{s,l}(r) \leq \Sigma_{u,k,l}\right] \\
&\leq \sum_{l=\frac{1}{2}\log^3(s-1)}^{s} \exp\left(-\frac{l}{2}\left(\Delta - \sqrt{\frac{\log^2(s-1)}{2l}}\right)^2\right) \\
&\leq t\exp\left(-\frac{1}{4}\log^3(w(t))\left(\Delta - \sqrt{\frac{\log^2 t}{\log^3(w(t))}}\right)^2\right) \\
&= t\exp\left(-\frac{1}{4}\log^2 t\right).
\end{aligned}$$

Therefore,

$$\mathbb{P}\left[\text{(B.9) is false}\right] \leq UK\exp\left(-\frac{1}{4}\log^2 t + 2\log t\right) + o\left(\frac{UK}{t^3}\right) = o\left(\frac{UK}{t^3}\right).$$

This proves the result for Q-ThS. □

For any time t, let

$$B_u(t) := \min\{s \geq 0 : Q_u(t-s) = 0\}$$

denote the time elapsed since the beginning of the current regenerative cycle for queue u. Alternately, at any time t, $t - B_u(t)$ is the last time instant at which queue u was zero.

The following lemma gives an upper bound on the sample-path queue-regret in terms of the number of sub-optimal schedules in the current regenerative cycle.

Lemma B.3. *For any $t \geq 1$,*

$$Q_u(t) - Q_u^*(t) \leq \sum_{l=t-B_u(t)+1}^{t} \left(\mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l) \right).$$

Proof. If $B_u(t) = 0$, i.e., if $Q_u(t) = 0$, then the result is trivially true.

Consider the case where $B_u(t) > 0$. Since $Q_u(l) > 0$ for all $t - B_u(t) + 1 \leq l \leq t$, we have

$$Q_u(l) = Q_u(l-1) + A_u(l) - S_u(l) \quad \forall t - B_u(t) + 1 \leq l \leq t.$$

This implies that

$$Q_u(t) = \sum_{l=t-B_u(t)+1}^{t} A_u(l) - S_u(l).$$

Moreover,

$$Q_u^*(t) = \max_{1 \leq s \leq t} \left(Q_u^*(0) + \sum_{l=s}^{t} A_u(l) - S_u^*(l) \right)^+ \geq \sum_{l=t-B_u(t)+1}^{t} A_u(l) - S_u^*(l).$$

Combining the above two expressions, we have

$$\begin{aligned}
Q_u(t) - Q_u^*(t) &\leq \sum_{l=t-B_u(t)+1}^{t} S_u^*(l) - S_u(l) \\
&= \sum_{l=t-B_u(t)+1}^{t} \sum_{k \in [K]} \left(R_{uk_u^*}(l) - R_{uk}(l) \right) \left(\mathsf{E}_{uk}(l) + \mathsf{I}_{uk}(l) \right) \\
&\leq \sum_{l=t-B_u(t)+1}^{t} \sum_{k \neq k_u^*} \left(\mathsf{E}_{uk}(l) + \mathsf{I}_{uk}(l) \right) \\
&\leq \sum_{l=t-B_u(t)+1}^{t} \left(\mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l) \right),
\end{aligned}$$

where the second inequality follows from the assumption that the service provided by each of the links is bounded by 1, and the last inequality from the fact that $\sum_{k \in [K]} \mathsf{E}_{uk}(l) = \mathsf{E}(l)\ \forall l, \forall u \in [U]$. □

In the next lemma, we derive a coarse high probability upper bound on the queue-length. This bound on the queue-length is used later to obtain a first cut bound on the length of the regenerative cycle in Lemma B.5.

Lemma B.4. *For any* $l \in [1, t]$,

$$\mathbb{P}\left[Q_u(l) > 2Kw(t)\right] = O\left(\frac{UK}{t^3}\right)$$

$\forall t$ *s.t.* $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$.

Proof. From Lemma B.3,

$$Q_u(t) - Q_u^*(t) \leq \sum_{l=t-B_u(t)+1}^{t} \left(\mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l)\right) \leq \sum_{l=1}^{t} \left(\mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l)\right).$$

Since $Q_u^*(t)$ is distributed according to $\pi_{(\lambda_u, \mu_u^*)}$,

$$\mathbb{P}\left[Q_u^*(t) > w(t)\right] = \frac{\lambda_u}{\mu_u^*}\left(\frac{\lambda_u\left(1-\mu_u^*\right)}{\left(1-\lambda_u\right)\mu_u^*}\right)^{w(t)} \leq \exp\left(w(t)\log\left(\frac{\lambda_u\left(1-\mu_u^*\right)}{\left(1-\lambda_u\right)\mu_u^*}\right)\right) \leq \frac{1}{t^3}$$

if $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$. The last inequality follows from the following bound –

$$\begin{aligned}
\log\left(\frac{\left(1-\lambda_u\right)\mu_u^*}{\lambda_u\left(1-\mu_u^*\right)}\right) &= \log\left(1 + \frac{\epsilon_u}{\lambda_u\left(1-\mu_u^*\right)}\right) \\
&\geq \log\left(1+4\epsilon_u\right) \quad \text{since } \left(\lambda_u\left(1-\mu_u^*\right) < 1/4\right) \\
&\geq \frac{3}{2}\epsilon_u.
\end{aligned}$$

Moreover, from Lemma B.1, we have

$$\mathbb{P}\left[\sum_{l=1}^{t} \mathsf{E}(l) > Kw(t)\right] = o\left(\frac{1}{t^3}\right).$$

Now, note that

$$\sum_{l=1}^{t}\sum_{k \neq k_u^*} \mathsf{I}_{uk}(l) \leq (K-1)w(t) + \sum_{l=w(t)+1}^{t}\sum_{k \neq k_u^*} \mathsf{I}_{uk}(l).$$

Therefore,

$$\mathbb{P}\left[\sum_{l=1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) > (K-1)w(t)\right] \leq \mathbb{P}\left[\sum_{l=w(t)+1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) > 0\right] = O\left(\frac{UK}{t^3}\right)$$

from Lemma B.2. Using the inequalities above, we have

$$\begin{aligned}\mathbb{P}\left[Q_u(t) > 2Kw(t)\right] &\leq \mathbb{P}\left[Q_u^*(t) > w(t)\right] + \mathbb{P}\left[\sum_{l=1}^{t}\mathsf{E}(l) > Kw(t)\right] \\ &\quad + \mathbb{P}\left[\sum_{l=1}^{t}\sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) > (K-1)w(t)\right] \\ &\leq \frac{1}{t^3} + O\left(\frac{UK}{t^3}\right) \\ &= O\left(\frac{UK}{t^3}\right).\end{aligned}$$

□

Lemma B.5. *Let $v_u'(t) = \frac{6K}{\epsilon_u}w(t)$ and let v_u be an arbitrary function. Then,*

$$\mathbb{P}\left[B_u\left(t - v_u(t)\right) > v_u'(t)\right] = O\left(\frac{UK}{t^3}\right)$$

$\forall t$ s.t. $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$ and $v_u(t) + v_u'(t) \leq t/2$.

Proof. Let $r(t) := t - v_u(t)$. Consider the events

$$Q_u(r(t) - v_u'(t)) \leq 2Kw(t), \tag{B.10}$$

$$\sum_{l=r(t)-v_u'(t)+1}^{r(t)} A_u(l) - R_{uk_u^*}(l) \leq -\frac{\epsilon_u}{2}v_u'(t), \tag{B.11}$$

$$\sum_{l=r(t)-v_u'(t)+1}^{r(t)} \mathsf{E}(l) + \sum_{k\neq k_u^*}\mathsf{I}_{uk}(l) \leq Kw(t). \tag{B.12}$$

By the definition of $v'_u(t)$,

$$2Kw(t) - \frac{\epsilon_u}{2} v'_u(t) \leq -Kw(t).$$

Given Events (B.10)-(B.12), the above inequality implies that

$$\begin{aligned} Q_u(r(t) - v'_u(t)) + \sum_{l=r(t)-v'_u(t)+1}^{r(t)} A_u(l) &\leq \sum_{l=r(t)-v'_u(t)+1}^{r(t)} R_{uk^*_u}(l) - \left(\mathsf{E}(l) + \sum_{k \neq k^*_u} \mathsf{I}_{uk}(l) \right) \\ &\leq \sum_{l=r(t)-v'_u(t)+1}^{r(t)} S_u(l), \end{aligned}$$

which further implies that $Q_u(l) = 0$ for some $l \in [r(t) - v'_u(t) + 1, r(t)]$. This gives us that $B_u(r(t)) \leq v'_u(t)$.

We now show that each of the events (B.10)-(B.12) occur with high probability. Consider the event (B.11) and note that $A_u(l) - R_{uk^*_u}(l)$ are i.i.d. random variables with mean $-\epsilon_u$ and bounded between -1 and 1. Using Chernoff bound for sum of bounded i.i.d. random variables, we have

$$\mathbb{P}\left[\sum_{l=r(t)-v'_u(t)+1}^{r(t)} A_u(l) - R_{uk^*_u}(l) > -\frac{\epsilon_u}{2} v'_u(t) \right] \leq \exp\left(-\frac{\epsilon_u^2}{8} v'_u(t) \right) \leq \frac{1}{t^3}$$

since $v'_u(t) \geq \frac{6K}{\epsilon_u} w(t) \geq \frac{24}{\epsilon_u^2} \log t$.

By Lemmas B.4, B.2 and B.1, the probability that any of the events (B.10), (B.12) does not occur is $O\left(\frac{UK}{t^3}\right)$ $\forall t$ s.t. $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$ and $v_u(t) + v'_u(t) \leq t/2$, and therefore we have the required result. □

Using the preceding upper bound on the regenerative cycle-length, we derive tighter bounds on the queue-length and the regenerative cycle-length in Lemmas B.7 and B.8 respectively. The following lemma is a useful intermediate result.

Lemma B.6. *For any $u \in [U]$ and t_2 s.t. $1 \le t_2 \le t$,*

$$\mathbb{P}\left[\max_{1\le s\le t_2}\left\{\sum_{l=t_2-s+1}^{t_2} A_u(l) - R_{uk_u^*}(l)\right\} \ge \frac{2\log t}{\epsilon_u}\right] \le \frac{1}{t^3}.$$

Proof. Let $X_s = \sum_{l=t_2-s+1}^{t_2} A_u(l) - R_{uk_u^*}(l)$. Since X_s is the sum of s i.i.d. random variables with mean ϵ_u and is bounded within $[-1, 1]$, Hoeffding's inequality gives

$$\begin{aligned}
\mathbb{P}\left[X_s \ge \frac{2\log t}{\epsilon_u}\right] &= \mathbb{P}\left[X_s - \mathbb{E}X_s \ge \epsilon_u s + \frac{2\log t}{\epsilon_u}\right] \\
&\le \exp\left(-\frac{2\left(\epsilon_u s + \frac{2\log t}{\epsilon_u}\right)^2}{4s}\right) \\
&\le \exp\left(-4\log t\right),
\end{aligned}$$

where the last inequality follows from the fact that $(a+b)^2 > 4ab$ for any $a, b \ge 0$. Using union bound over all $1 \le s \le t_2$ gives the required result. □

Lemma B.7. *Let $v_u'(t) = \frac{6K}{\epsilon_u} w(t)$ and v_u be an arbitrary function. Then,*

$$\mathbb{P}\left[Q_u(t - v_u(t)) > \left(\frac{2}{\epsilon_u} + 5\right)\log t + 30K\frac{v_u'(t)\log^2 t}{t}\right] = O\left(\frac{UK}{t^3}\right)$$

$\forall t$ s.t. $\frac{w(t)}{\log t} \ge \frac{2}{\epsilon_u}$ and $v_u(t) + v_u'(t) \le t/2$.

Proof. Let $r(t) = t - v_u(t)$. Now, consider the events

$$B_u(r(t)) \le v_u'(t), \tag{B.13}$$

$$\sum_{l=r(t)-s+1}^{r(t)} A_u(l) - R_{uk_u^*}(l) \le \frac{2\log t}{\epsilon_u}\, 1 \le s \le v_u'(t), \tag{B.14}$$

$$\sum_{l=r(t)-v'_u(t)+1}^{r(t)} \mathsf{E}(l) + \sum_{k\neq k^*_u} \mathsf{I}_{uk}(l) \leq 5\log t + 5K\left(\log^3\left(r(t)\right) - \log^3\left(r(t)-v'_u(t)\right)\right). \tag{B.15}$$

Given the above events, we have

$$\begin{aligned}
Q_u(r(t)) &= \sum_{l=r(t)-B_u(r(t))+1}^{r(t)} A_u(l) - S(l) \\
&\leq \sum_{l=r(t)-B_u(r(t))+1}^{r(t)} A_u(l) - R_{uk^*_u}(l) + \mathsf{E}(l) + \sum_{k\neq k^*} \mathsf{I}_{uk}(l) \\
&\leq \left(\frac{2}{\epsilon_u}+5\right)\log t + 5K\left(\log^3\left(r(t)\right) - \log^3\left(r(t)-v'_u(t)\right)\right) \\
&\leq \left(\frac{2}{\epsilon_u}+5\right)\log t + 15K\frac{v'_u(t)\log^2 t}{(r(t)-v'_u(t))} \\
&\leq \left(\frac{2}{\epsilon_u}+5\right)\log t + 30K\frac{v'_u(t)\log^2 t}{t},
\end{aligned}$$

where the last inequality is true if $v_u(t)+v'_u(t)\leq t/2$. From Lemmas B.5, B.6, B.2 and B.1, probability of each the events (B.13)-(B.15) is $1-O\left(\frac{UK}{t^3}\right)$ and therefore, we have the required result. □

Lemma B.8. *Let* $v'_u(t)=\frac{6K}{\epsilon_u}w(t)$ *and* $v_u(t)=\frac{24\log t}{\epsilon_u^2}+\frac{60K}{\epsilon_u}\frac{v'_u(t)\log^2 t}{t}$. *Then,*

$$\mathbb{P}\left[B_u(t) > v_u(t)\right] = O\left(\frac{UK}{t^3}\right)$$

$\forall t$ *s.t.* $\frac{w(t)}{\log t}\geq\frac{2}{\epsilon_u}$ *and* $v_u(t)+v'_u(t)\leq t/2$.

Proof. Let $r(t)=t-v_u(t)$. As in Lemma B.5, consider the events

$$Q_u(r(t)) \leq \left(\frac{2}{\epsilon_u}+5\right)\log t + 30K\frac{v'_u(t)\log^2 t}{t}, \tag{B.16}$$

$$\sum_{l=r(t)+1}^{t} A_u(l) - R_{uk_u^*}(l) \leq -\frac{\epsilon_u}{2} v_u(t), \tag{B.17}$$

$$\sum_{l=r(t)+1}^{t} \mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l) \leq 5 \log t + 5K \left(\log^3 t - \log^3 \left(r(t) \right) \right). \tag{B.18}$$

The definition of $v_u(t)$ and events (B.16)-(B.18) imply that

$$\begin{aligned} Q_u(r(t)) + \sum_{l=r(t)+1}^{t} A_u(l) &\leq \sum_{l=r(t)+1}^{t} R_{uk_u^*}(l) - \sum_{l=r(t)+1}^{t} \mathsf{E}(l) + \sum_{k \neq k_u^*} \mathsf{I}_{uk}(l) \\ &\leq \sum_{l=r(t)+1}^{t} S_u(l), \end{aligned}$$

which further implies that $Q(l) = 0$ for some $l \in [r(t)+1, t]$ and therefore $B_u(t) \leq v_u(t)$. We can again show that each of the events (B.16)-(B.18) occurs with high probability. Particularly, by Lemmas B.1, B.2 and B.7, the probability that any one of the events (B.16), (B.18) does not occur is $O\left(\frac{UK}{t^3}\right)$ $\forall t$ s.t. $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$ and $v_u(t) + v_u'(t) \leq t/2$. We can bound the probability of event (B.17) in the same way as event (B.14) in Lemma B.5 to show that it occurs with probability at least $\frac{1}{t^3}$. Combining all these gives us the required high probability result. □

Proof of Theorem 3.2. The proof is based on two main ideas: one is that the regenerative cycle length is not very large, and the other is that the algorithm has correctly identified the optimal matching in late stages. We combine Lemmas B.2

and B.8 to bound the regret at any time t s.t. $\frac{w(t)}{\log t} \geq \frac{2}{\epsilon_u}$ and $v_u(t) + v'_u(t) \leq t/2$:

$$\begin{aligned}
\Psi_u(t) &= \mathbb{E}\left[Q_u(t) - Q^*_u(t)\right] \\
&\leq \mathbb{E}\left[Q_u(t) - Q^*_u(t) \middle| B_u(t) \leq v_u(t)\right] \mathbb{P}\left[B_u(t) \leq v_u(t)\right] \\
&\quad + \mathbb{E}\left[Q_u(t) - Q^*_u(t) \middle| B_u(t) > v_u(t)\right] \mathbb{P}\left[B_u(t) > v_u(t)\right] \\
&\leq \mathbb{E}\left[\sum_{l=t-v_u(t)+1}^{t} \mathsf{E}(l) + \sum_{k \neq k^*_u} \mathsf{I}_{uk}(l)\right] + t\mathbb{P}\left[B_u(t) > v_u(t)\right] \qquad \text{(B.19)} \\
&\leq K\left(\log^3(t) - \log^3(t - v_u(t))\right) \qquad \text{(B.20)} \\
&\quad + t\mathbb{P}\left[\sum_{l=t-v_u(t)+1}^{t} \sum_{k \neq k^*_u} \mathsf{I}_{uk}(l) > 0\right] + t\mathbb{P}\left[B_u(t) > v_u(t)\right] \qquad \text{(B.21)} \\
&\leq 3K \log^2 t \log\left(1 + \frac{v_u(t)}{t - v_u(t)}\right) + O\left(\frac{UK}{t^2}\right) \\
&= O\left(K \frac{v_u(t) \log^2 t}{t - v_u(t)}\right) + O\left(\frac{U}{tw(t)}\right) \\
&= O\left(K \frac{v_u(t) \log^2 t}{t}\right),
\end{aligned}$$

where (B.19) follows from Lemma B.3, and the last two terms in inequality (B.21) are bounded using Lemmas B.2 and B.8. □

Proof of Corollary 3.1. We first note the following:

(i) $\frac{t}{w(t)} \geq \frac{24K}{\epsilon_u}$ implies that $v'_u(t) \leq \frac{t}{4}$,

(ii) $\frac{t}{w(t)} \geq 15K^2 \log t$ implies that $\frac{24}{\epsilon_u^2} \log t \geq \frac{60K}{\epsilon_u} \frac{v'_u(t) \log^2 t}{t}$, and therefore $v_u(t) \leq \frac{48}{\epsilon_u^2} \log t$

(iii) $\frac{t}{\log t} \geq \frac{198}{\epsilon_u^2}$ implies that $v_u(t) \leq \frac{t}{4}$.

These inequalities when applied to Theorem 3.2 give the required result. □

B.2.2 Lower Bounds for α-Consistent Policies

As mentioned earlier, we prove asymptotic and early stage lower bounds for a class of policies called the α-consistent class (Definition 3.1). In order to prove Theorems 3.1 and 3.3, we use techniques from existing work in the MAB literature along with some new lower bounding ideas specific to queueing systems. Specifically, we use lower bounds for α-consistent policies on the expected number of times a sub-optimal server is scheduled. This lower bound, shown (in Lemma B.10) specifically for the problem of scheduling a unique optimal matching, is similar in style to the traditional bandit lower bound by Lai et al. [91] but holds in the non-asymptotic setting. Also, as opposed the traditional change of measure proof technique used in [91], the proof (similar to the more recent ones [33, 114, 41]) uses results from hypothesis testing (Lemma B.9).

Lemma B.9 ([133])**.** *Consider two probability measures P and Q, both absolutely continuous with respect to a given measure. Then for any event $\mathcal{A}$ we have:*

$$P(\mathcal{A}) + Q(\mathcal{A}^c) \geq \frac{1}{2}\exp\{-\min(\mathrm{KL}(P||Q), \mathrm{KL}(Q||P))\}.$$

Proof. Let $p = P(\mathcal{A})$ and $q = Q(\mathcal{A}^c)$. From standard properties of KL divergence we have that,

$$\mathrm{KL}(P||Q) \geq \mathrm{KL}(p, q)$$

Therefore, it is sufficient to prove that

$$p + q \geq \frac{1}{2}\exp\left(-p\log\frac{p}{1-q} - (1-p)\log\frac{1-p}{q}\right) = \frac{1}{2}\left(\frac{1-q}{p}\right)^p\left(\frac{q}{1-p}\right)^{1-p}.$$

Now,

$$\begin{aligned}
\left(\frac{1-q}{p}\right)^{p}\left(\frac{q}{1-p}\right)^{1-p} &= \left(\sqrt{\frac{1-q}{p}}\right)^{2p}\left(\sqrt{\frac{q}{1-p}}\right)^{2(1-p)} \\
&\leq \left(\frac{1}{2}\left(2p\cdot\sqrt{\frac{1-q}{p}}+2(1-p)\cdot\sqrt{\frac{q}{1-p}}\right)\right)^{2} \\
&= \left(\sqrt{p(1-q)}+\sqrt{q(1-p)}\right)^{2} \\
&\leq 2(p(1-q)+q(1-p)) \\
&< 2(p+q)
\end{aligned}$$

as required. □

Lemma B.10. *For any problem instance $(\boldsymbol{\lambda},\boldsymbol{\mu})$ and any α-consistent policy, there exist constants τ and C s.t. for any $u\in[U]$, $k\neq k^*_u$ and $t>\tau$,*

$$\begin{aligned}
&\mathbb{E}\left[T_{uk}(t+1)\right]+\sum_{u'\neq u}\mathbb{1}\left\{k^*_{u'}=k\right\}\mathbb{E}\left[T_{u'k^*_u}(t+1)\right] \\
&\geq \frac{1}{\mathrm{KL}\left(\mu_{min},\frac{\mu_{max}+1}{2}\right)}\left((1-\alpha)\log t-\log(4KC)\right).
\end{aligned}$$

Proof. Without loss of generality, let the optimal servers for the U queues be denoted by the first U indices. In other words, a server $k>U$ is not an optimal server for any queue, i.e., for any $u'\in[U]$, $K\geq k>U$, $\mathbb{1}\left\{k^*_{u'}=k\right\}=0$. Also, let $\beta=\frac{\mu_{max}+1}{2}$.

We will first consider the case $k\leq U$. For a fixed user u and server $k\leq U$, let u' be the queue that has k as the best server, i.e., $k^*_{u'}=k$. Now consider the two problem instances $(\boldsymbol{\lambda},\boldsymbol{\mu})$ and $(\boldsymbol{\lambda},\hat{\boldsymbol{\mu}})$, where $\hat{\boldsymbol{\mu}}$ is the same as $\boldsymbol{\mu}$ except for the two entries corresponding to indices $(u,k),(u',k^*_u)$ replaced by β. Therefore, for the problem instance $(\boldsymbol{\lambda},\hat{\boldsymbol{\mu}})$, the best servers are swapped for queues u and u'

and remain the same for all the other queues. Let $\mathbb{P}^t_{\boldsymbol{\mu}}$ and $\mathbb{P}^t_{\hat{\boldsymbol{\mu}}}$ be the distributions corresponding to the arrivals, chosen servers and rates obtained in the first t plays for the two instances under a fixed α-consistent policy. Recall that $T_{uk}(t+1) = \sum_{s=1}^{t} \mathbb{1}\{\kappa_u(s) = k\}\ \forall u \in [U], k \in [K]$. Define the event $\mathcal{A} = \{T_{uk}(t+1) > t/2\}$. By the definition of α-consistency there exists a fixed integer τ and a fixed constant C such that for all $t > \tau$ we have,

$$\mathbb{E}^t_{\boldsymbol{\mu}}\left[\sum_{s=1}^{t} \mathbb{1}\{\kappa_u(s) = k\}\right] \leq Ct^{\alpha}$$
$$\mathbb{E}^t_{\hat{\boldsymbol{\mu}}}\left[\sum_{s=1}^{t} \mathbb{1}\{\kappa_u(s) = k'\}\right] \leq Ct^{\alpha}\ ,\ \forall k' \neq k.$$

A simple application of Markov's inequality yields

$$\mathbb{P}^t_{\boldsymbol{\mu}}(\mathcal{A}) \leq \frac{2C}{t^{1-\alpha}}$$
$$\mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\mathcal{A}^c) \leq \frac{2C(K-1)}{t^{1-\alpha}}.$$

We can now use Lemma B.9 to conclude that

$$\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}) \geq (1-\alpha)\log t - \log(4KC). \tag{B.22}$$

It is, therefore, sufficient to show that

$$\mathrm{KL}\left(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}\right) = \mathrm{KL}\left(\mu_{uk}, \beta\right) \mathbb{E}^t_{\boldsymbol{\mu}}[T_{uk}(t+1)] + \mathrm{KL}\left(\mu_{u'k^*_u}, \beta\right) \mathbb{E}^t_{\boldsymbol{\mu}}[T_{u'k^*_u}(t+1)].$$

For the sake of brevity we write the scheduling sequence in the first t time-slots $\{\boldsymbol{\kappa}(1), \boldsymbol{\kappa}(2), ..., \boldsymbol{\kappa}(t)\}$ as $\boldsymbol{\kappa}^{(t)}$, and similarly we define $\mathbf{A}^{(t)}$ as the number of arrivals to the queue and $\mathbf{S}^{(t)}$ as the service offered by the scheduled servers in the first t

time-slots. Let $\mathbf{Z}^{(t)} = (\boldsymbol{\kappa}^{(t)}, \mathbf{A}^{(t)}, \mathbf{S}^{(t)})$. The KL-divergence term can now be written as

$$\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}) = \mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}}(\mathbf{Z}^{(t)}) || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\mathbf{Z}^{(t)})).$$

We can apply the chain rule of divergence to conclude that

$$\begin{aligned}\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}}(\mathbf{Z}^{(t)}) || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\mathbf{Z}^{(t)})) &= \mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}}(\mathbf{Z}^{(t-1)}) || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\mathbf{Z}^{(t-1)})) \\ &\quad + \mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}}(\boldsymbol{\kappa}(t) \mid \mathbf{Z}^{(t-1)}) || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\boldsymbol{\kappa}(t) \mid \mathbf{Z}^{(t-1)})) \\ &\quad + \mathbb{E}^t_{\boldsymbol{\mu}} \left[\mathbb{1}\{\kappa_u(t) = k\} \mathrm{KL}\left(\mu_{uk}, \beta\right) + \mathbb{1}\{\kappa_{u'}(t) = k^*_u\} \mathrm{KL}\left(\mu_{u'k^*_u}, \beta\right) \right].\end{aligned}$$

We can apply this iteratively to obtain

$$\begin{aligned}\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}) &= \sum_{s=1}^{t} \mathbb{E}^t_{\boldsymbol{\mu}} \left[\mathbb{1}\{\kappa_u(s) = k\} \mathrm{KL}\left(\mu_{uk}, \beta\right) \right] \\ &\quad + \sum_{s=1}^{t} \mathbb{E}^t_{\boldsymbol{\mu}} \left[\mathbb{1}\{\kappa_{u'}(s) = k^*_u\} \mathrm{KL}\left(\mu_{u'k^*_u}, \beta\right) \right] \\ &\quad + \sum_{l=1}^{t} \mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}}(\boldsymbol{\kappa}(l) \mid \mathbf{Z}^{(l-1)}) || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}(\boldsymbol{\kappa}(l) \mid \mathbf{Z}^{(l-1)})) \qquad \text{(B.23)}\end{aligned}$$

Note that the second summation in (B.23) is zero, as over a sample path the policy pulls the same servers irrespective of the parameters. Therefore, we obtain

$$\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}) = \mathrm{KL}\left(\mu_{uk}, \beta\right) \mathbb{E}^t_{\boldsymbol{\mu}}[T_{uk}(t+1)] + \mathrm{KL}\left(\mu_{u'k^*_u}, \beta\right) \mathbb{E}^t_{\boldsymbol{\mu}}[T_{u'k^*_u}(t+1)],$$

which can be substituted in (B.22) to obtain the required result for $K \leq U$.

Now, consider the case $k > U$, where $\sum_{u \in U} \mathbb{1}\{k^*_u = k\} = 0$. We again compare the two problem instances $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and $(\boldsymbol{\lambda}, \hat{\boldsymbol{\mu}})$, where $\hat{\boldsymbol{\mu}}$ is the same as $\boldsymbol{\mu}$ except for the entry corresponding to index (u, k) replaced by β. Therefore, for the

problem instance $(\boldsymbol{\lambda}, \hat{\boldsymbol{\mu}})$, the best server for user u is server k while the best servers for all other queues remain the same. We can again use the same technique as before to obtain

$$\mathrm{KL}(\mathbb{P}^t_{\boldsymbol{\mu}} || \mathbb{P}^t_{\hat{\boldsymbol{\mu}}}) = \mathrm{KL}\left(\mu_{uk}, \beta\right) \mathbb{E}^t_{\boldsymbol{\mu}}[T_{uk}(t+1)],$$

which, along with (B.22), gives the required result for $K > U$. □

As a corollary of the above result, we now derive lower bound on the total expected number of sub-optimal schedules summed across all queues. In addition, we also show, for each individual queue, a lower bound for those servers which are sub-optimal for all the queues. As in the proof of Lemma B.10, we assume without loss of generality that the first U indices denote the optimal servers for the U queues.

Corollary B.1. *For any problem instance $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ and any α-consistent policy, there exist constants τ and C s.t. for any $t > \tau$,*

(a)

$$2\Delta \sum_{u \in [U]} \sum_{k \neq k^*_u} \mathbb{E}\left[T_{uk}(t+1)\right] \geq U(K-1) D(\boldsymbol{\mu}) \left((1-\alpha)\log t - \log(4KC)\right),$$

(b) for any $u \in [U]$,

$$2\Delta \sum_{k \neq k^*_u} \mathbb{E}\left[T_{uk}(t+1)\right] \geq (U-1) D(\boldsymbol{\mu}) \left((1-\alpha)\log t - \log(4KC)\right),$$

(c) and for any $u \in [U]$,

$$\Delta \sum_{k > U} \mathbb{E}\left[T_{uk}(t+1)\right] \geq (K-U) D(\boldsymbol{\mu}) \left((1-\alpha)\log t - \log(4KC)\right),$$

where $D(\boldsymbol{\mu})$ is given by (3.10).

Proof. To prove part (a), we observe that a unique optimal server for each queue in the system implies that

$$\sum_{u\in[U]}\sum_{k\neq k_u^*}\mathbb{E}\left[T_{uk}(t+1)\right] \geq \sum_{u\in[U]}\sum_{u'\neq u}\mathbb{E}\left[T_{uk_{u'}^*}(t+1)\right]$$
$$= \sum_{u\in[U]}\sum_{k\neq k_u^*}\sum_{u'\neq u}\mathbb{1}\left\{k_{u'}^*=k\right\}\mathbb{E}\left[T_{u'k_u^*}(t+1)\right].$$

Now, from Lemma B.10, there exist constants C and τ such that for $t>\tau$,

$$2\sum_{u\in[U]}\sum_{k\neq k_u^*}\mathbb{E}\left[T_{uk}(t+1)\right] \geq \sum_{u\in[U]}\sum_{k\neq k_u^*}\left(\mathbb{E}\left[T_{uk}(t+1)\right]+\sum_{u'\neq u}\mathbb{1}\left\{k_{u'}^*=k\right\}\mathbb{E}\left[T_{u'k_u^*}(t+1)\right]\right)$$
$$\geq \frac{U(K-1)}{\mathrm{KL}\left(\mu_{min},\frac{\mu_{max}+1}{2}\right)}\left((1-\alpha)\log t-\log(4KC)\right).$$

Using the definition of $D(\boldsymbol{\mu})$ in the above inequality gives part (a) of the corollary.

To prove part (b), we can assume without loss of generality that a perfect matching is scheduled in every time-slot. Using this, and the fact that any server is assigned to at most one queue in every time-slot, for any $u\in[U]$, we have

$$T_{uk_u^*}(t+1)+\sum_{k\neq k_u^*}T_{uk}(t+1)=t\geq T_{uk_u^*}(t+1)+\sum_{u'\neq u}T_{u'k_u^*}(t+1),$$

which gives us

$$\sum_{k\neq k_u^*}T_{uk}(t+1)\geq\max\left\{\sum_{u'\neq u}T_{uk_{u'}^*}(t+1),\sum_{u'\neq u}T_{u'k_u^*}(t+1)\right\}. \tag{B.24}$$

From Lemma B.10 we have, for any $u'\neq u$ and for $t>\tau$,

$$\mathbb{E}\left[T_{uk_{u'}^*}(t+1)\right]+\mathbb{E}\left[T_{u'k_u^*}(t+1)\right]\geq\frac{1}{\mathrm{KL}\left(\mu_{min},\frac{\mu_{max}+1}{2}\right)}\left((1-\alpha)\log t-\log(4KC)\right),$$

which gives

$$\sum_{u' \neq u} \mathbb{E}\left[T_{uk^*_{u'}}(t+1)\right] + \mathbb{E}\left[T_{u'k^*_u}(t+1)\right] \geq \frac{U-1}{\mathrm{KL}\left(\mu_{min}, \frac{\mu_{max}+1}{2}\right)} \left((1-\alpha)\log t - \log(4KC)\right).$$

Combining the above with (B.24), we have for $t > \tau$

$$\begin{aligned}\sum_{k \neq k^*_u} \mathbb{E}\left[T_{uk}(t+1)\right] &\geq \max\left\{\sum_{u' \neq u} \mathbb{E}\left[T_{uk^*_{u'}}(t+1)\right], \sum_{u' \neq u} \mathbb{E}\left[T_{u'k^*_u}(t+1)\right]\right\} \\ &\geq \frac{U-1}{2\mathrm{KL}\left(\mu_{min}, \frac{\mu_{max}+1}{2}\right)} \left((1-\alpha)\log t - \log(4KC)\right).\end{aligned}$$

To prove part (c), we use the fact that $\mathbb{1}\left\{k^*_{u'} = k\right\} = 0$ for any $u' \in [U]$, $K \geq k > U$. Therefore, for $t > \tau$, we have

$$\begin{aligned}\sum_{k>U} \mathbb{E}\left[T_{uk}(t+1)\right] &= \sum_{k>U}\left(\mathbb{E}\left[T_{uk}(t+1)\right] + \sum_{u' \neq u} \mathbb{1}\left\{k^*_{u'} = k\right\} \mathbb{E}\left[T_{u'k^*_u}(t+1)\right]\right) \\ &\geq \frac{K-U}{\mathrm{KL}\left(\mu_{min}, \frac{\mu_{max}+1}{2}\right)} \left((1-\alpha)\log t - \log(4KC)\right),\end{aligned}$$

which gives the required result. □

B.2.2.1 Late Stage: Proof of Theorem 3.1

The following lemma, which gives a lower bound on the queue-regret in terms of probability of sub-optimal schedule in a single time-slot, is the key result used in the proof of Theorem 3.1. The proof for this lemma is based on the idea that the growth in regret in a single-time slot can be lower bounded in terms of the probability of sub-optimal schedule in that time-slot.

Lemma B.11. *For any problem instance characterized by* $(\boldsymbol{\lambda}, \boldsymbol{\mu})$*, and for any schedul-*

ing policy, and user $u \in [U]$,

$$\Psi_u(t) \geq \lambda_u \sum_{k \neq k_u^*} \Delta_{uk} \mathbb{P}\left[\mathbb{1}\{\kappa_u(t) = k\} = 1\right].$$

Proof. For the given queueing system, consider an alternate coupled queueing system such that

1. the two systems start with the same initial condition,
2. the arrival process for both the systems is the same, and
3. the service process for the alternate system is independent of the arrival process and i.i.d. across time-slots. For each queue in the alternate system, the service offered by different servers at any time-slot could possibly be dependent on each other but has the same marginal distribution as that in the original system and is independent of the service offered to other queues.

We first show that, under any scheduling policy, the regret for the alternate system has the same distribution as that for the original system. Note that the evolution of the queues is a function of the process $(\mathbf{Z}(l))_{l \geq 1} := (\mathbf{A}(l), \boldsymbol{\kappa}(l), \mathbf{S}(l))_{l \geq 1}$. To prove that this process has the same distribution in both the systems, we use induction on the size of the finite-dimensional distribution of the process. In other words, we show that the distribution of the vector $(\mathbf{Z}(l))_{l=1}^{t}$ is the same for the two systems for all t by induction on t.

Suppose that the hypothesis is true for $t-1$. Now consider the conditional distribution of $\mathbf{Z}(t)$ given $(\mathbf{Z}(l))_{l=1}^{t-1}$. Given $(\mathbf{Z}(l))_{l=1}^{t-1}$, the distribution of $(\mathbf{A}(t), \boldsymbol{\kappa}(t))$

is identical for the two systems for any scheduling policy since the two systems have the same arrival process. Also, given $\left((\mathbf{Z}(l))_{l=1}^{t-1}, \mathbf{A}(t), \boldsymbol{\kappa}(t)\right)$, the distribution of $\mathbf{S}(t)$ depends only on the marginal distribution of the scheduled servers given by $\boldsymbol{\kappa}(t)$ which is again the same for the two systems. Therefore, $(\mathbf{Z}(l))_{l=1}^{t}$ has the same distribution in the two systems. Since the statement is true for $t = 1$, it is true for all t.

Thus, to lower bound the queue-regret for any queue $u \in [U]$ in the original system, it is sufficient to lower bound the corresponding queue-regret of an alternate queueing system constructed as follows: let $\{U(t)\}_{t\geq 1}$ be i.i.d. random variables distributed uniformly in $(0, 1)$. For the alternate system, let the service process for queue u and server k be given by $R_{uk}(t) = \mathbb{1}\{U(t) \leq \mu_{uk}\}$. Since $\mathbb{E}[R_{uk}(t)] = \mu_{uk}$, the marginals of the service offered by each of the servers is the same as the original system. In addition, the initial condition, the arrival process and the service process for all other queues in the alternate system are identical to those in the original system.

We now lower bound the queue-regret for queue u in the alternate system. Note that, since $\mu_u^* > \mu_{uk}\ \forall k \neq k_u^*$, we have $R_{uk_u^*}(t) \geq R_{uk}(t)\ \forall k \neq k_u^*, \forall t$. This implies that $Q_u^*(t) \leq Q_u(t)\ \forall t$. Now, for any given t, using the fact that $Q_u^*(t-1) \leq Q_u(t-1)$, it is easy to see that

$$Q_u(t) - Q_u^*(t) \geq \mathbb{1}\{A_u(t) = 1\}\left(R_{k_u^*}(t) - \sum_{k=1}^{K} \mathbb{1}\{\kappa_u(t) = k\} R_{uk}(t)\right).$$

Therefore,

$$\begin{aligned}\mathbb{E}\left[Q_u(t) - Q_u^*(t)\right] &\geq \mathbb{E}\left[\mathbb{1}\{A_u(t)=1\}\left(R_{k_u^*}(t) - \sum_{k=1}^{K}\mathbb{1}\{\kappa_u(t)=k\}R_{uk}(t)\right)\right] \\ &= \lambda_u \sum_{k\neq k_u^*}\mathbb{P}\left[\mathbb{1}\{\kappa_u(t)=k\}=1\right]\mathbb{P}\left[\mu_{uk} < U(t) \leq \mu_u^*\right] \\ &= \lambda_u \sum_{k\neq k_u^*}\Delta_{uk}\mathbb{P}\left[\mathbb{1}\{\kappa_u(t)=k\}=1\right].\end{aligned}$$

□

We now use Lemma B.11 in conjunction with the lower bound for the expected number of sub-optimal schedules for an α-consistent policy (Corollary B.1) to prove Theorem 3.1.

Proof of Theorem 3.1. From Lemma B.11 we have,

$$\begin{aligned}\Psi_u(t) &\geq \lambda_u \sum_{k\neq k_u^*}\Delta_{uk}\mathbb{P}\left[\mathbb{1}\{\kappa_u(t)=k\}=1\right] \\ &\geq \lambda_{min}\Delta \sum_{k\neq k_u^*}\mathbb{P}\left[\mathbb{1}\{\kappa_u(t)=k\}=1\right]. \qquad \text{(B.25)}\end{aligned}$$

Therefore,

$$\sum_{s=1}^{t}\sum_{u\in[U]}\Psi_u(s) \geq \lambda_{min}\Delta \sum_{u\in[U]}\sum_{k\neq k_u^*}\mathbb{E}\left[T_{uk}(t+1)\right].$$

We now claim that

$$\sum_{u\in[U]}\Psi_u(t) \geq \frac{U(K-1)}{8t}\lambda_{min}D(\boldsymbol{\mu})(1-\alpha) \qquad \text{(B.26)}$$

for infinitely many t. This follows from part (a) of Corollary B.1 and the following fact:

Fact B.1. *For any bounded sequence* $\{a_n\}$*, if there exist constants* C *and* n_0 *such that* $\sum_{m=1}^{n} a_m \geq C \log n \; \forall n \geq n_0$*, then* $a_n \geq \frac{C}{2n}$ *infinitely often.*

Similarly, for any $u \in U$, it follows from parts (b) and (c) of Corollary B.1 that

$$\Psi_u(t) \geq \frac{\max\{U-1, 2(K-U)\}}{8t} \lambda_{min} D(\boldsymbol{\mu})(1-\alpha) \tag{B.27}$$

for infinitely many t. □

B.2.2.2 Early Stage: Proof of Theorem 3.3

In order to prove Theorem 3.3, we first derive, in the following lemma, a lower bound on the queue-regret in terms of the expected number of sub-optimal schedules.

Lemma B.12. *For any system with parameters* $(\boldsymbol{\lambda}, \boldsymbol{\mu})$*, any policy, and any user* $u \in [U]$*, the regret is lower bounded by*

$$\Psi_u(t) \geq \sum_{k \neq k_u^*} \Delta_{uk} \mathbb{E}\left[T_{uk}(t+1)\right] - \epsilon_u t.$$

Proof. Since $Q_u(0) \sim \pi_{\lambda_u, \mu_u^*}$, we have,

$$\begin{aligned}
\Psi_u(t) &= \mathbb{E}\left[Q_u(t) - Q_u^*(t)\right] \\
&= \mathbb{E}\left[Q_u(t) - Q_u(0)\right] \\
&\geq \mathbb{E}\left[\sum_{l=1}^{t} A_u(l) - S_u(l)\right] \\
&= \lambda_u t - \sum_{k=1}^{K} \mathbb{E}\left[T_{uk}(t+1)\right] \mu_{uk} \\
&= \lambda_u t - \left(t - \sum_{k \neq k_u^*} \mathbb{E}\left[T_{uk}(t+1)\right]\right) \mu *_u - \sum_{k \neq k_u^*} \mathbb{E}\left[T_{uk}(t+1)\right] \mu_{uk} \\
&= \sum_{k \neq k_u^*} \Delta_{uk} \mathbb{E}\left[T_{uk}(t+1)\right] - \epsilon_u t.
\end{aligned}$$

□

We now use this lower bound along with the lower bound on the expected number of sub-optimal schedules for α-consistent policies (Corollary B.1).

Proof of Theorem 3.3. To prove part (a) of the theorem, we use Lemma B.12 and part (a) of corollary B.1 as follows: For any $\gamma > \frac{1}{1-\alpha}$, there exist constants C_1 and τ such that for all $t \in [\max\{C_1 K^\gamma, \tau\}, (K-1)\frac{D(\boldsymbol{\mu})}{4\bar{\epsilon}}]$,

$$\begin{aligned}
\frac{1}{U} \sum_{u \in [U]} \Psi_u(t) &\geq \frac{\Delta}{U} \sum_{u \in [U]} \left(\sum_{k \neq k_u^*} \mathbb{E}\left[T_{uk}(t+1)\right] - \epsilon_u t \right) \\
&\geq (K-1)\frac{D(\boldsymbol{\mu})}{2}\left((1-\alpha)\log t - \log(KC_1)\right) - \bar{\epsilon} t \\
&\geq (K-1)\frac{D(\boldsymbol{\mu})}{2}\frac{\log t}{\log\log t} - \bar{\epsilon} t \\
&\geq (K-1)\frac{D(\boldsymbol{\mu})}{4}\frac{\log t}{\log\log t},
\end{aligned}$$

where the last two inequalities follow since $t \geq C_1 K^\gamma$ and $t \leq (K-1)\frac{D(\boldsymbol{\mu})}{4\bar{\epsilon}}$.

Part (b) of the theorem can be similarly shown using parts (b) and (c) of corollary B.1. $\square$

Appendix C

Proofs and Additional Results in Chapter 4

C.0.3 Proof of Theorem 4.1

Proof of Theorem 4.1. Consider an ergodic Markov policy $\varphi \in \mathfrak{M}$ such that $\boldsymbol{\lambda} \in \Lambda^{\varphi}(\boldsymbol{\mu})$. We use the notation $\mathbb{P}_{\pi}$ to denote probabilities corresponding to the stationary distribution under policy φ. Let

$$P_{j',j} = \mathbb{P}_{\pi}\left[\mathbf{J}(t) = j \mid \mathbf{J}(t-1) = j'\right] \quad \forall j', j \in \mathcal{J},$$
$$\sigma_j = \mathbb{P}_{\pi}\left[\mathbf{J}(t) = j\right] \quad \forall j \in \mathcal{J},$$

and

$$\alpha_{\mathbf{r}}(j,h) = \mathbb{P}_{\pi}\left[\mathbf{S}(t) = \mathbf{r} \mid \mathbf{J}(t) = j, H(t) = h\right]$$
$$\forall \mathbf{r} \in \mathcal{R}(j,h),\ \forall j \in \mathcal{J}, h \in \mathcal{H}.$$

It is easy to verify that $\mathbf{P} \in \mathcal{W}_{|\mathcal{J}|}$, $\boldsymbol{\sigma} = \boldsymbol{\sigma}\mathbf{P}$,

$$\mathsf{C}_0 \sum_{j',j \in \mathcal{J}} \sigma_{j'} P_{j',j} \| \left(j' - j\right)^{+} \|_1 + \mathsf{C}_1 \sum_{j \in \mathcal{J}} \sigma_j \|j\|_1 = C^{\varphi}(\boldsymbol{\mu}, \boldsymbol{\lambda}),$$

and

$$\sum_{j \in \mathcal{J}} \sigma_j \sum_{h \in \mathcal{H}} \mu(h) \sum_{\mathbf{r} \in \mathcal{R}(j,h)} \alpha_{\mathbf{r}}(j,h)\mathbf{r} = \mathbb{E}_{\pi}\left[\mathbf{S}(t)\right].$$

Since $\boldsymbol{\lambda} \in \Lambda^{\varphi}(\boldsymbol{\mu})$, we have $\boldsymbol{\lambda} < \mathbb{E}_{\pi}\left[\mathbf{S}(t)\right]$. Therefore, for

$$\varphi' := \left(\mathbf{P}, \boldsymbol{\alpha} = \{\boldsymbol{\alpha}(j,h)\}_{j \in \mathcal{J}, h \in \mathcal{H}}\right),$$

we have $\varphi' \in \mathfrak{MG}$, $\boldsymbol{\lambda} \in \Lambda^{\varphi'}(\boldsymbol{\mu})$ and $C^{\varphi'}(\boldsymbol{\mu}, \boldsymbol{\lambda}) = C^{\varphi}(\boldsymbol{\mu}, \boldsymbol{\lambda})$. □

C.0.4 Proofs of Lemmas 4.1 and 4.2

Proof of Lemma 4.1. Let $\mathcal{F}$ and $\mathcal{D}$ denote the feasible sets of L and its dual respectively. By Theorem 2 in [139], to prove (I), it is sufficient to prove that $\mathcal{F}$ and $\mathcal{D}$ are continuous multifunctions on $\mathbb{R}^d \times \mathcal{S}$. The feasible set of the linear program depends only on $(\boldsymbol{\mu}, \boldsymbol{\lambda})$ and not on $\mathbf{c}$. By Proposition 6 in [139], $\mathcal{F}$ is continuous on $\mathcal{S}$ if

(i) the dimension of $\mathcal{F}$ is constant on $\mathcal{S}$, and

(ii) for any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$, there exists a neighborhood $\mathcal{V}$ of $(\boldsymbol{\mu}, \boldsymbol{\lambda})$ such that, if a particular inequality constraint is tight (satisfied with equality) for all $x \in \mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\lambda})$, then for any $(\boldsymbol{\mu}', \boldsymbol{\lambda}') \in \mathcal{V}$, the corresponding constraint is tight for all $x \in \mathcal{F}(\boldsymbol{\mu}', \boldsymbol{\lambda}')$.

The above two conditions are satisfied if

(i) the equality constraints are the same for every $\mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\lambda})$, and

(ii) for any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$, no inequality constraint is tight for every $x \in \mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\lambda})$.

These can be verified to be true for all $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$. Therefore, $\mathcal{F}$ is continuous on $\mathcal{S}$.

According to Corollary 11 in [139], $\mathcal{D}$ is continuous on $\mathbb{R}^d \times \mathcal{S}$ if $\mathcal{F}$ is bounded. This is again true since any feasible solution is a set of probability mass functions. Therefore, by Theorem 2 in [139], C^* is continuous on $\mathbb{R}^d \times \mathcal{S}$.

To prove (II), i.e., that the optimal solution set $\mathcal{O}^*_{\mathbf{c}}(\cdot)$ is continuous on $\mathcal{U}_{\mathbf{c}}$, we first note that $\mathcal{O}^*_{\mathbf{c}}(\boldsymbol{\mu}, \boldsymbol{\lambda})$ is the feasible set for a set of linear constraints which is

same as that for $\mathcal{F}(\boldsymbol{\mu}, \boldsymbol{\lambda})$ in addition to the equality constraint

$$\mathbf{c} \cdot (\boldsymbol{\sigma}, \boldsymbol{\beta}) = C^*_{\mathbf{c}}(\boldsymbol{\mu}, \boldsymbol{\lambda}).$$

By definition, the set $\mathcal{O}^*_{\mathbf{c}}(\boldsymbol{\mu}, \boldsymbol{\lambda})$ is non-empty for any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$ and is a singleton for any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}}$. Now consider any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}}$. Using (I) and Theorem 3.1 in [44], the extreme point set of $\mathcal{O}^*_{\mathbf{c}}$ is continuous at $(\boldsymbol{\mu}, \boldsymbol{\lambda})$. Since $\mathcal{O}^*_{\mathbf{c}}$ is convex and is a singleton at $(\boldsymbol{\mu}, \boldsymbol{\lambda})$, we have that $\mathcal{O}^*_{\mathbf{c}}$ is continuous at $(\boldsymbol{\mu}, \boldsymbol{\lambda})$. □

Proof of Lemma 4.2. For any $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{S}$, $(\boldsymbol{\mu}, \boldsymbol{\lambda}) \in \mathcal{U}_{\mathbf{c}^{\epsilon_p}}$ if the vector $\mathbf{c}^{\epsilon_p}$ is not perpendicular to any of the faces of the polytope given by the feasible set of $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda})$. For any $1 \leq k \leq d-1$, consider any k-dimensional face of this polytope. The probability that the vector $\mathbf{c}^{\epsilon_p}$ lies in the $d-k$ dimensional space orthogonal to this face is zero. Since there are only a finite number of faces, by the union bound, we have

$$\mathbb{P}\left[(\boldsymbol{\mu}, \boldsymbol{\lambda}) \notin \mathcal{U}_{\mathbf{c}^{\epsilon_p}} \mid \mathbf{J}(0), \mathbf{Q}(1)\right] = 0.$$

□

C.0.5 Proof of Theorem 4.2

We use continuity of the linear program L (Lemma 4.1) to prove part 1 of Theorem 4.2. To prove stability (part 2 of Theorem 4.2), we show that the long term Lyapunov drift is negative.

C.0.5.1 Cost Optimality

Part 1 of the theorem follows easily from the continuity of the optimal value of the linear program L. The expected cost at time t under policy $\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g, \epsilon_s)$

is given by

$$\begin{aligned}
&\mathbb{E}_{\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)}\left[C(t) \,\middle|\, \mathbf{J}(0),\mathbf{Q}(1)\right] \\
&= \sum_{j',j\in\mathcal{J}} \mathbb{P}\left[\mathbf{J}(t-1)=j'\right] \mathbf{P}(\boldsymbol{\sigma}^*,\epsilon_s)_{j',j} \left(\mathsf{C}_0\|\left(j'-j\right)^+\|_1 + \mathsf{C}_1\|j\|_1\right).
\end{aligned}$$

Since $\mathbf{J}(0)$ is chosen according to the static-split rule given by $\boldsymbol{\sigma}^*$, we have

$$\mathbb{P}\left[\mathbf{J}(t-1)=j'\right] = \sigma^*_{j'}.$$

This gives us

$$\begin{aligned}
\mathbb{E}_{\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)}\left[C(t) \,\middle|\, \mathbf{J}(0),\mathbf{Q}(1)\right] &\leq (1-\epsilon_s)\sum_{j'\in\mathcal{J}} \sigma^*_{j'}\mathsf{C}_1\|j'\|_1 + \epsilon_s\left(M\mathsf{C}_0 + \sum_{j\in\mathcal{J}}\sigma^*_j\mathsf{C}_1\|j\|_1\right) \\
&\leq C^*_{\mathbf{c}^0}\left(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g\right) + M\mathsf{C}_0\epsilon_s \\
&\leq C^*_{\mathbf{c}^0}\left(\boldsymbol{\mu},\boldsymbol{\lambda}\right) + \kappa\epsilon_s + \gamma(\epsilon_g),
\end{aligned}$$

for some increasing function $\gamma(\cdot)$ such that $\lim_{\epsilon_g\to 0}\gamma(\epsilon_g)=0$. This follows from the continuity of $C^*_{\mathbf{c}^0}\left(\boldsymbol{\mu},\cdot\right)$ (part (I) of Lemma 4.1). Therefore,

$$\begin{aligned}
\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}\left[C(t) \,\middle|\, \mathbf{J}(0),\mathbf{Q}(1)\right] &\leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu},\boldsymbol{\lambda}) + \kappa\epsilon_s + \gamma(\epsilon_g) \\
&\leq C^{\mathfrak{M}}(\boldsymbol{\mu},\boldsymbol{\lambda}) + \kappa\epsilon_s + \gamma(\epsilon_g),
\end{aligned}$$

where the last inequality follows from (4.6). This proves part 1 of Theorem 4.2.

C.0.5.2 Stability: Negative Lyapunov Drift

We show stability in the sense of Definition 4.1 by first showing that the quadratic Lyapunov drift for the policy $\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)$ is negative outside a finite set. Let $V(\mathbf{q}) := \sum_{m,u} q^2_{m,u}$ be the Lyapunov function. For any $T>0$, $t>0$,

let $\Delta_T(t) := V(\mathbf{Q}(t+T)) - V(\mathbf{Q}(t))$ be the T-step Lyapunov drift. The following lemma shows that the long term Lyapunov drift is negative outside a finite set:

Lemma C.1. *For any $\boldsymbol{\mu}, \boldsymbol{\lambda}$, there exists constants T, B such that for any $t \in \mathbb{N}$,*

$$\mathbb{E}_{\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)}\left[\Delta_T(t) \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t)\right] \leq BT - \epsilon_g T \sum_{m,u} Q_{m,u}(t). \tag{C.1}$$

Proof. Since $\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)$ has a single ergodic class, the marginal distribution of the Markov chain $\{\mathbf{J}(t)\}_{t\geq 0}$ converges to $\boldsymbol{\sigma}^*$. Moreover, we can choose a constant $T \in \mathbb{N}$ such that

$$\max_{j' \in \mathcal{J}} \sum_{l=1}^{T} \sum_{j \in \mathcal{J}} \left| \mathbb{P}\left[\mathbf{J}(l) = j \,\middle|\, \mathbf{J}(0) = j'\right] - \sigma_j^* \right| \leq \frac{T\epsilon_g}{2\bar{\mathsf{R}}}. \tag{C.2}$$

Using standard arguments for quadratic drift with bounded arrivals and service, there exists a constant B' such that,

$$\Delta_T(t) \leq B'T + 2\sum_{l=0}^{T-1} (\mathbf{A}(t+l) - \mathbf{S}(t+l)) \cdot \mathbf{Q}(t+l).$$

Let $\boldsymbol{\alpha}^*$ be the set of convex combinations related to the unique optimal solution $(\boldsymbol{\sigma}^*, \boldsymbol{\beta}^*)$ through (4.9). Since the policy $\varphi(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g, \epsilon_s)$ allocates rates according to the Max-Weight rule, we have

$$\begin{aligned}
\Delta_T(t) &\leq B'T + 2\sum_{l=0}^{T-1} (\mathbf{A}(t+l) - \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{J}(t+l), H(t+l))} \alpha_{\mathbf{r}}^*(\mathbf{J}(t+l), H(t+l))\mathbf{r}) \cdot \mathbf{Q}(t+l) \\
&\leq BT + 2\sum_{l=0}^{T-1} (\mathbf{A}(t+l) - \sum_{\mathbf{r} \in \mathcal{R}(\mathbf{J}(t+l), H(t+l))} \alpha_{\mathbf{r}}^*(\mathbf{J}(t+l), H(t+l))\mathbf{r}) \cdot \mathbf{Q}(t)
\end{aligned}$$

for $B = B' + \max\{\bar{\mathsf{A}}^2, \bar{\mathsf{R}}^2\}(T-1)$ by using the assumption that the arrivals and service per time-slot are bounded at every queue. Taking averages in the above

inequality, we get

$$\begin{aligned}
&\mathbb{E}_{\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)}\left[\Delta_T(t) \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t)\right] \\
&\leq BT + 2T\boldsymbol{\lambda}\cdot\mathbf{Q}(t) - 2\sum_{l=0}^{T-1}\mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))}\alpha^*_{\mathbf{r}}(\mathbf{J}(t+l),H(t+l))\mathbf{r} \,\middle|\, \mathbf{J}(t-1)\right]\cdot\mathbf{Q}(t).
\end{aligned} \tag{C.3}$$

Now, for any $l \in [0, T-1]$, let

$$y(t+l)_j := \mathbb{P}\left[\mathbf{J}(t+l) = j \,\middle|\, \mathbf{J}(t-1)\right].$$

By the choice of T given by (C.2), we have

$$\sum_{l=0}^{T-1}\|\mathbf{y}(t+l) - \boldsymbol{\sigma}^*\|_1\bar{\mathsf{R}} \leq T\epsilon_g/2.$$

Then

$$\begin{aligned}
&\sum_{l=0}^{T-1}\mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))}\alpha^*_{\mathbf{r}}(\mathbf{J}(t+l),H(t+l))\mathbf{r} \,\middle|\, \mathbf{J}(t-1)\right] \\
&= \sum_{l=0}^{T-1}\sum_{j\in\mathcal{J}} y(t+l)_j\left(\sum_{h\in\mathcal{H}}\mu(h)\sum_{\mathbf{r}\in\mathcal{R}(j,h)}\alpha^*_{\mathbf{r}}(j,h)\mathbf{r}\right) \\
&\geq \sum_{l=0}^{T-1}\left(\left(\sum_{j\in\mathcal{J}}\sigma^*_j\sum_{h\in\mathcal{H}}\mu(h)\sum_{\mathbf{r}\in\mathcal{R}(j,h)}\alpha^*_{\mathbf{r}}(j,h)\mathbf{r}\right) - \|\mathbf{y}(t+l) - \boldsymbol{\sigma}^*\|_1\bar{\mathsf{R}}\right) \\
&\geq T(\boldsymbol{\lambda} + \epsilon_g/2),
\end{aligned}$$

where the last inequality follows since any solution to the linear program $L_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g)$ satisfies its constraints

$$\sum_{j\in\mathcal{J}}\sigma^*_j\sum_{h\in\mathcal{H}}\mu(h)\sum_{\mathbf{r}\in\mathcal{R}(j,h)}\alpha^*_{\mathbf{r}}(j,h)\mathbf{r} \geq \boldsymbol{\lambda} + \epsilon_g.$$

Substituting this inequality in (C.3), we get the required result

$$\mathbb{E}_{\varphi(\boldsymbol{\mu},\boldsymbol{\lambda}+\epsilon_g,\epsilon_s)}\left[\Delta_T(t)\,\middle|\,\mathbf{J}(t-1),\mathbf{Q}(t)\right] \leq BT - T\epsilon_g \sum_{m,u} Q_{m,u}(t).$$

□

C.0.6 Proof of Theorem 4.3

As in the proof of Theorem 4.2, we use continuity of the linear program L (Lemma 4.1) to prove part 1 of Theorem 4.3. To prove stability (part 2 of Theorem 4.3), we show that the long term Lyapunov drift is negative outside a finite set given the event

$$\mathcal{E}^0 := (\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g) \in \mathcal{U}_{\mathbf{c}^{\epsilon_p}}. \tag{C.4}$$

This negative Lyapunov drift, as in the Foster's theorem for time-homogeneous Markov chains, is then used to prove stability as per Definition 4.1.

C.0.6.1 Cost Optimality

Let $\mathbf{y}(t)$ denote the distribution of $\mathbf{J}(t)$ and

$$\tilde{\mathbf{P}}(t) := \left(\mathbb{P}\left[\mathbf{J}(t) = j \,\middle|\, \mathbf{J}(t-1) = j'\right]\right)_{j',j\in\mathcal{J}}$$

denote the transition probability matrix at time t. Therefore, the expected cost at time t under policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$ is given by

$$\mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}\left[C(t)\,\middle|\,\mathbf{J}(0),\mathbf{Q}(1)\right] = \sum_{j',j\in\mathcal{J}} y(t-1)_{j'}\tilde{P}(t)_{j',j}\left(\mathsf{C}_0\|\left(j'-j\right)^+\|_1 + \mathsf{C}_1\|j\|_1\right). \tag{C.5}$$

Let $\hat{\boldsymbol{\mu}}(t)$, $\hat{\boldsymbol{\lambda}}(t)$ be the estimated statistics at the beginning of time-slot t. Then, under policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$, $\left(\mathbf{J}(t-1), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\lambda}}(t)\right)_{t\geq 0}$ is a time-inhomogeneous Markov chain. Given $\hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\lambda}}(t)$, the transition probability matrix $\mathbf{P}(t)$ at time t for the BS activation transitions is given by

$$\begin{aligned}\mathbf{P}(t) &:= \left(\mathbb{P}\left[\mathbf{J}(t) = j \;\middle|\; \mathbf{J}(t-1) = j', \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\lambda}}(t)\right]\right)_{j',j\in\mathcal{J}} \\ &= (1-\epsilon_l(t))\mathbf{P}(\hat{\boldsymbol{\sigma}}(t), \epsilon_s) + \epsilon_l(t)\mathbf{P_1},\end{aligned}$$

where $\mathbf{P_1}$ is the transition probability matrix that takes the activation state to $\mathbf{1}$ (all BSs active) with probability 1 from any state.

Given $\mathcal{E}^0$ as defined in (C.4), let $(\boldsymbol{\sigma}^*, \boldsymbol{\beta}^*) \in \mathcal{O}^*_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g)$ be the unique solution to the linear program $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g)$. Since $\lim_{t\to\infty}\left(\hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\lambda}}(t)\right) \overset{\text{a.s.}}{=} (\boldsymbol{\mu}, \boldsymbol{\lambda})$, from part (II) of Lemma 4.1 we have that $\lim_{t\to\infty}\hat{\boldsymbol{\sigma}}(t) \overset{\text{a.s.}}{=} \boldsymbol{\sigma}^*$ and $\lim_{t\to\infty}\tilde{\mathbf{P}}(t) = \lim_{t\to\infty}\mathbb{E}\left[\mathbf{P}(t)\right] = \mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)$. Using Lemma 4.3(b), we have $\lim_{t\to\infty}\mathbf{y}(t) = \boldsymbol{\sigma}^*$. Applying these along with Lemma 4.2 to (C.5) yields

$$\begin{aligned}\limsup_{t\to\infty}\mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}\left[C(t) \;\middle|\; \mathbf{J}(0), \mathbf{Q}(1)\right] &\leq C^*_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g) + M\mathsf{C}_0\epsilon_s \\ &\leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g) + \sqrt{|\mathcal{H}|+1}\mathsf{C}_1\epsilon_p + M\mathsf{C}_0\epsilon_s \\ &\leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g) + \kappa(\epsilon_p+\epsilon_s),\end{aligned}$$

for a constant κ that depends on the size of the network and C_0, C_1. Now, from the continuity of $C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \cdot)$ (part (I) of Lemma 4.1), we have

$$\begin{aligned}\limsup_{t\to\infty}\mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)}\left[C(t) \;\middle|\; \mathbf{J}(0), \mathbf{Q}(1)\right] &\leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}+\epsilon_g) + \kappa(\epsilon_p+\epsilon_s) \\ &\leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}) + \kappa(\epsilon_p+\epsilon_s) + \gamma(\epsilon_g),\end{aligned}$$

for some increasing function $\gamma(\cdot)$ such that $\lim_{\epsilon_g \to 0} \gamma(\epsilon_g) = 0$. This gives us

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)} \left[C(t) \,\middle|\, \mathbf{J}(0), \mathbf{Q}(1) \right] \leq C^*_{\mathbf{c}^0}(\boldsymbol{\mu}, \boldsymbol{\lambda}) + \kappa(\epsilon_p + \epsilon_s) + \gamma(\epsilon_g)$$
$$\leq C^{\mathfrak{M}}(\boldsymbol{\mu}, \boldsymbol{\lambda}) + \kappa(\epsilon_p + \epsilon_s) + \gamma(\epsilon_g),$$

where the last inequality follows from (4.6). This proves part 1 of Theorem 4.3.

C.0.6.2 Stability: Negative Lyapunov Drift

Similar to Theorem 4.2, we show that the long term Lyapunov drift is negative outside a finite set. But unlike in Theorem 4.2, this negative drift condition holds only after a random time that has bounded second moment.

Lemma C.2. *For any $\boldsymbol{\mu}, \boldsymbol{\lambda}$, there exists constants T, B and a random time Γ such that $\mathbb{E}\left[\Gamma^2 \mid \mathbf{J}(0), \mathbf{Q}(1)\right] < \infty$, and for any $t > \Gamma$,*

$$\mathbb{E}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)} \left[\Delta_T(t) \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t), \mathbf{J}(0), \mathbf{Q}(1), \Gamma, \mathcal{E}^0 \right] \leq BT - \frac{\epsilon_g}{2} T \sum_{m,u} Q_{m,u}(t). \tag{C.6}$$

Similar to the proof of Foster's theorem, we show below that (C.6) implies stability as per Definition 4.1.

Lemma C.3. *If the condition given by* (C.6) *is satisfied, then for any $b > 0$,*

$$\lim_{t\to\infty} \frac{1}{t} \sum_{l=1}^{t} \mathbb{P}_{\phi(\epsilon_p,\epsilon_s,\epsilon_g)} \left[\mathbf{Q}(l) \in \mathcal{A} \,\middle|\, \mathbf{J}(0), \mathbf{Q}(1) \right] > \frac{b}{\bar{B}},$$

where $\bar{B} = B + b$, $\bar{Q} = \frac{2\bar{B}}{\epsilon_g}$ and $\mathcal{A} = \left\{ \mathbf{Q} \in \mathbb{R}^{M\times n} : \sum_{m,u} Q_{m,u} < \bar{Q} \right\}$. Therefore, the network is stable under policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$.

Proof. For ease of notation, we do not explicitly write the conditioning on $\mathbf{J}(0), \mathbf{Q}(1)$. Let the condition given by (C.6) be true. Then under policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$, we have from (C.6) that

$$\mathbb{E}\left[\Delta_T(t) \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t), \Gamma, \mathcal{E}^0\right] \leq -bT + \bar{B}T\mathbb{1}\left\{\mathbf{Q}(t) \in \mathcal{A}\right\},$$

for any $t > \Gamma$. Now, let $I^* := \min\{i : (i-1)T \geq \Gamma\}$. Consider, for any $k \in \mathbb{N}$,

$$\begin{aligned}\sum_{l=1}^{T} \mathbb{E}\left[V(\mathbf{Q}(kT+l)) - V(\mathbf{Q}(l))\right] = \sum_{l=1}^{T} &\mathbb{E}\left[V(\mathbf{Q}(kT+l) - V(\mathbf{Q}((I^*-1)T+l))\right] \\ &+ \mathbb{E}\left[V(\mathbf{Q}((I^*-1)T+l)) - V(\mathbf{Q}(l))\right]. \quad \text{(C.7)}\end{aligned}$$

Now,

$$\begin{aligned}&\sum_{l=1}^{T} \mathbb{E}\left[V(\mathbf{Q}(kT+l)) - V(\mathbf{Q}((I^*-1)T+l))\right] \\ &= \sum_{l=1}^{T} \mathbb{E}\left[\sum_{i=I^*-1}^{k-1} \Delta_T(iT+l)\right] \\ &= \sum_{l=1}^{T} \mathbb{E}\left[\sum_{i=I^*-1}^{k-1} \mathbb{E}\left[\Delta_T(iT+l) \mid \mathbf{Q}(iT+l), \Gamma, \mathcal{E}^0\right]\right] \quad \text{(C.8)} \\ &\leq \sum_{l=1}^{T} \mathbb{E}\left[\sum_{i=I^*-1}^{k-1} \left(-bT + \bar{B}T\mathbb{1}\left\{\mathbf{Q}(iT+l) \in \mathcal{A}\right\}\right)\right] \\ &\leq -b(k-1)T^2 + bT\mathbb{E}\left[\Gamma\right] + \bar{B}T\mathbb{E}\left[\sum_{t=\Gamma+1}^{kT} \mathbb{1}\left\{\mathbf{Q}(t) \in \mathcal{A}\right\}\right].\end{aligned}$$

In (C.8), we have used that $\mathbb{P}\left[\mathcal{E}^0\right] = 1$ from Lemma 4.2.

Moreover, since $(I^*-1)T < \Gamma + T$, we have

$$\mathbf{Q}((I^*-1)T+l) \leq \mathbf{Q}(l) + \bar{\mathsf{A}}(\Gamma+T)$$

for any $l \in \mathbb{N}$, which gives

$$\begin{aligned}
&\sum_{l=1}^{T} \mathbb{E}\left[V(\mathbf{Q}((I^*-1)T+l)) - V(\mathbf{Q}(l))\right] \\
&\leq \sum_{l=1}^{T} \mathbb{E}\left[\sum_{m,u}\left((\bar{\mathsf{A}}(\Gamma+T)+Q_{m,u}(l))^2 - Q_{m,u}(l)^2\right)\right] \\
&\leq \mathbb{E}\left[nMT(\bar{\mathsf{A}}(\Gamma+T))^2 + 2\sum_{l=1}^{T}\sum_{m,u}\bar{\mathsf{A}}(\Gamma+T)Q_{m,u}(l)\right] \\
&\leq \mathbb{E}\left[nMT(\bar{\mathsf{A}}(\Gamma+T))^2\right] + \mathbb{E}\left[2\bar{\mathsf{A}}(\Gamma+T)\sum_{l=1}^{T}\sum_{m,u}\left(\bar{\mathsf{A}}(l-1)+Q_{m,u}(1)\right)\right] \\
&\leq T\mathbb{E}\left[nM\bar{\mathsf{A}}^2(\Gamma^2+3T\Gamma+2T^2)\right] + T\mathbb{E}\left[2\bar{\mathsf{A}}(\Gamma+T)\sum_{m,u}Q_{m,u}(1)\right].
\end{aligned}$$

Applying the above two inequalities in (C.7) and rearranging the terms, we get

$$\bar{B}\mathbb{E}\left[\sum_{t=\Gamma+1}^{kT}\mathbb{1}\left\{\mathbf{Q}(t)\in\mathcal{A}\right\}\right] \geq bkT - \mathbb{E}\left[Y\right] + \frac{1}{T}\sum_{l=1}^{T}\mathbb{E}\left[V(\mathbf{Q}(kT+l)) - V(\mathbf{Q}(l))\right],$$

where

$$Y = nM\bar{\mathsf{A}}^2\Gamma^2 + \left(3nM\bar{\mathsf{A}}^2T + 2\bar{\mathsf{A}}\sum_{m,u}Q_{m,u}(1) + b\right)\Gamma + 2T^2 + 2\bar{\mathsf{A}}T\sum_{m,u}Q_{m,u}(1) + bT.$$

We have $\mathbb{E}\left[Y\right] < \infty$ since $\mathbb{E}\left[\Gamma^2\right] < \infty$. In addition, we also have

$$\limsup_{k\to\infty}\frac{1}{kT}\mathbb{E}\left[\frac{1}{T}\sum_{l=1}^{T}V(\mathbf{Q}(l))\right] \leq \limsup_{k\to\infty}\frac{1}{kT}\mathbb{E}\left[\frac{1}{T}\sum_{l=1}^{T}V(\bar{\mathsf{A}}(l-1)+\mathbf{Q}(1))\right] = 0.$$

Therefore,

$$\begin{aligned}
\bar{B}\liminf_{k\to\infty}\frac{1}{kT}\sum_{t=1}^{kT}\mathbb{P}\left[\mathbf{Q}(t)\in\mathcal{A}\right] &\geq \bar{B}\liminf_{k\to\infty}\frac{1}{kT}\mathbb{E}\left[\sum_{t=\Gamma+1}^{kT}\mathbb{1}\left\{\mathbf{Q}(t)\in\mathcal{A}\right\}\right] \\
&\geq b - \limsup_{k\to\infty}\frac{1}{kT}\mathbb{E}\left[Y + \frac{1}{T}\sum_{l=1}^{T}V(\mathbf{Q}(l))\right],
\end{aligned}$$

which gives us the required result

$$\liminf_{k\to\infty} \frac{1}{kT} \sum_{t=1}^{kT} \mathbb{P}\left[\mathbf{Q}(t) \in \mathcal{A} \,\middle|\, \mathbf{J}(0), \mathbf{Q}(1)\right] \geq \frac{b}{\bar{B}}.$$

□

Before we proceed to prove Lemma C.2, we will prove an intermediate result which shows that the transition probability matrices used by the policy to select the activation vector converge to a matrix that is close to the optimal. This result along with Lemma 4.3 allows us to show that the distribution of the activation vector converges to the optimal invariant distribution.

For any $k > 0$, let $\tilde{\boldsymbol{\mu}}(k)$, $\tilde{\boldsymbol{\lambda}}(k)$ denote the empirical distributions of channels and the empirical means of arrivals respectively obtained from the first k explore samples. For every $t > 0$, $\delta > 0$, define the events

$$\begin{aligned}
\mathcal{E}_l(t) &:= \left\{ \sum_{s=1}^{t} E_l(s) \geq \frac{1}{2} \log^2 t \right\}, \\
\mathcal{E}_{\tilde{\mu}}(t,\delta) &:= \left\{ \|\tilde{\boldsymbol{\mu}}(t) - \boldsymbol{\mu}\|_1 \leq \delta \right\}, \\
\mathcal{E}_{\tilde{\lambda}}(t,\delta) &:= \left\{ \|\tilde{\boldsymbol{\lambda}}(t) - \boldsymbol{\lambda}\|_1 \leq \delta \right\},
\end{aligned}$$

and

$$\mathcal{E}(t,\delta) := \mathcal{E}_l(t) \bigcap_{k \geq \frac{1}{2}\log^2 t} \left\{ \mathcal{E}_{\tilde{\mu}}(k,\delta') \cap \mathcal{E}_{\tilde{\lambda}}(k,\delta') \right\},$$

where $\delta' = \frac{1}{2}\min(\delta, \epsilon_g/\bar{\mathsf{R}})$.

Lemma C.4. *For any $\delta > 0$, let $T_1(\delta) := \min\{t : \mathcal{E}(t,\delta)$ is true$\}$. Then,*

$$\mathbb{E}\left[T_1(\delta)^2 \mid \mathbf{J}(0), \mathbf{Q}(1)\right] < \infty.$$

Proof. For ease of notation, we do not explicitly write the conditioning on $\mathbf{J}(0), \mathbf{Q}(1)$.

Consider the mean number of explore samples in the first t slots.

$$\begin{aligned}\mathbb{E}\left[\sum_{s=1}^{t} E_l(s)\right] &= \sum_{s=2}^{t} \frac{2\log s}{s} \\ &\geq \int_{s=e}^{t+1} \frac{2\log s}{s}\, \mathsf{d}s \\ &= \log^2(t+1) - 1 \\ &\geq \frac{3}{4}\log^2(t),\end{aligned}$$

for all $t \geq 3$. Using the Chernoff bound for Bernoulli random variables, we have $\forall t \geq 3$,

$$\mathbb{P}\left[\mathcal{E}_l(t)^c\right] \leq \exp\left(-\frac{1}{32}\log^2 t\right).$$

Using the Hoeffding's inequality for Bernoulli random variables, for any k, $\epsilon > 0$,

$$\begin{aligned}\mathbb{P}\left[\mathcal{E}_{\tilde{\lambda}}(k,\epsilon)^c\right] &\leq \sum_{u=1}^{n} \mathbb{P}\left[|\tilde{\lambda}_u(k) - \lambda_u| \geq \frac{1}{n}\epsilon\right] \\ &\leq n\exp\left(-2k\left(\frac{\epsilon}{n\bar{\mathsf{A}}}\right)^2\right).\end{aligned}$$

In addition, using Pinsker's inequality it can be shown [137] that for any k, $\epsilon > 0$,

$$\mathbb{P}\left[\mathcal{E}_{\tilde{\mu}}(k,\epsilon)^c\right] \leq (k+1)^{|\mathcal{H}|}\exp\left(-\frac{\epsilon^2}{2}k\right).$$

Now, let $\delta' = \frac{1}{2}\min(\delta, \epsilon_g/\bar{\mathsf{R}})$. Using the above inequalities, we have $\forall t \geq 3$,

$$\begin{aligned}\mathbb{P}\left[T_1(\delta) > t\right] &\leq \mathbb{P}\left[\mathcal{E}(t,\delta)^c\right] \\ &\leq \mathbb{P}\left[\mathcal{E}_l(t)^c\right] + \sum_{k=\frac{1}{2}\log^2 t}^{\infty} \left(\mathbb{P}\left[\mathcal{E}_{\tilde{\mu}}(k,\delta')^c\right] + \mathbb{P}\left[\mathcal{E}_{\tilde{\lambda}}(k,\delta')^c\right]\right) \\ &= o\left(\frac{1}{t^3}\right),\end{aligned}$$

which gives us

$$\mathbb{E}\left[T_1(\delta)^2\right] = 2\sum_{t=0}^{\infty} t\mathbb{P}\left[T_1(\delta) > t\right] < \infty.$$

□

Given $\mathcal{E}^0$, let $(\boldsymbol{\sigma}^*, \boldsymbol{\beta}^*) \in \mathcal{O}^*_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$ be the unique solution to the linear program $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$. Due to the continuity of the solution set of $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$ given $\mathcal{E}^0$ (from Lemma 4.1), there exists a positive function $\delta_1 \mapsto f(\delta_1)$ such that, if $(\boldsymbol{\mu}', \boldsymbol{\lambda}' + \epsilon_g) \in \mathcal{S}$ and

$$\|\boldsymbol{\mu}' - \boldsymbol{\mu}\|_1 + \|\boldsymbol{\lambda}' - \boldsymbol{\lambda}\|_1 \leq f(\delta_1),$$

then for any $(\boldsymbol{\sigma}', \boldsymbol{\beta}') \in \mathcal{O}^*_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}', \boldsymbol{\lambda}' + \epsilon_g)$,

$$\|\boldsymbol{\sigma}' - \boldsymbol{\sigma}^*\|_1 \leq \delta_1.$$

The following lemma shows that continuity of the solution of $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$ implies convergence of the activation vector transition probability matrices.

Lemma C.5. *If $(\boldsymbol{\mu}, \boldsymbol{\lambda} + 2\epsilon_g) \in \mathcal{S}$, then for any δ_1, t such that $\epsilon_l(t) \leq \delta_1/4$, the event $\mathcal{E}^0 \cap \mathcal{E}(t, f(\delta_1/2))$ implies the event*

$$\tilde{\mathcal{E}}(t, \delta_1) := \left\{\|\mathbf{P}(l) - \mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)\|_1 \leq \delta_1 \ \forall l > t\right\}.$$

Proof. The event $\mathcal{E}(t, f(\delta_1/2))$ implies that for any $l > t$, $(\hat{\boldsymbol{\mu}}(l), \hat{\boldsymbol{\lambda}}(l) + \epsilon_g) \in \mathcal{S}$ and

$$\|\hat{\boldsymbol{\mu}}(l) - \boldsymbol{\mu}\|_1 + \|\hat{\boldsymbol{\lambda}}(l) - \boldsymbol{\lambda}\|_1 \leq f(\delta_1/2).$$

Therefore, we have

$$\|\hat{\boldsymbol{\sigma}}(l) - \boldsymbol{\sigma}^*\|_1 \leq \delta_1/2,$$

which gives us

$$\begin{aligned}\|\mathbf{P}(l) - \mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)\|_1 &\leq \|\mathbf{P}(\hat{\boldsymbol{\sigma}}(l), \epsilon_s) - \mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)\|_1 + 2\epsilon_l(l) \\ &\leq \|\hat{\boldsymbol{\sigma}}(l) - \boldsymbol{\sigma}^*\|_1 + \delta_1/2 \\ &\leq \delta_1.\end{aligned}$$

□

We now prove the negative Lyapunov drift condition (Lemma C.2).

Proof of Lemma C.2. There exists a constant B' such that,

$$\Delta_T(t) \leq B'T + 2\sum_{l=0}^{T-1}(\mathbf{A}(t+l) - \mathbf{S}(t+l)) \cdot \mathbf{Q}(t+l).$$

Assume $\mathcal{E}^0$ is true and let $\boldsymbol{\alpha}^*$ be the set of convex combinations related to the unique optimal solution $(\boldsymbol{\sigma}^*, \boldsymbol{\beta}^*)$ through (4.9). Since the policy $\phi(\epsilon_p, \epsilon_s, \epsilon_g)$ allocates rates according to the Max-Weight rule, we have

$$\begin{aligned}\Delta_T(t) &\leq B'T + 2\sum_{l=0}^{T-1}(\mathbf{A}(t+l) - \sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))} \alpha^*_{\mathbf{r}}(\mathbf{J}(t+l), H(t+l))\mathbf{r}) \cdot \mathbf{Q}(t+l) \\ &\leq BT + 2\sum_{l=0}^{T-1}(\mathbf{A}(t+l) - \sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))} \alpha^*_{\mathbf{r}}(\mathbf{J}(t+l), H(t+l))\mathbf{r}) \cdot \mathbf{Q}(t)\end{aligned}$$

for $B = B' + \max\{\bar{\mathsf{A}}^2, \bar{\mathsf{R}}^2\}(T-1)$ by using the assumption that the arrivals and service per time-slot are bounded at every queue. To prove (C.6), it is sufficient to

prove that there exists a $T > 0$ and a random time Γ such that $\mathbb{E}\left[\Gamma^2 \mid \mathbf{J}(0), \mathbf{Q}(1)\right] < \infty$, and for any $t > \Gamma$,

$$Z \geq T(\boldsymbol{\lambda} + \epsilon_g/4), \tag{C.9}$$

where Z is given by (C.10).

$$Z := \sum_{l=0}^{T-1} \mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))} \alpha^*_{\mathbf{r}}(\mathbf{J}(t+l), H(t+l))\mathbf{r} \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t), \mathbf{J}(0), \mathbf{Q}(1), \Gamma, \mathcal{E}^0\right] \tag{C.10}$$

$$= \sum_{l=0}^{T-1} \mathbb{E}\left[\mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))} \alpha^*_{\mathbf{r}}(\mathbf{J}(t+l), H(t+l))\mathbf{r} \,\middle|\, Y(t), \mathcal{E}^0\right] \,\middle|\, \mathbf{J}(t-1), \Gamma, \mathcal{E}^0\right], \tag{C.11}$$

where $Y(t) = (\mathbf{J}(t-1), \hat{\boldsymbol{\mu}}(t), \hat{\boldsymbol{\lambda}}(t), \Gamma)$.

This can be justified as follows: if (C.9) is true, then for any $t > \Gamma$,

$$\begin{aligned}\mathbb{E}\left[\Delta_T(t) \,\middle|\, \mathbf{J}(t-1), \mathbf{Q}(t), \mathbf{J}(0), \mathbf{Q}(1), \Gamma, \mathcal{E}^0\right] &\leq BT + 2T\boldsymbol{\lambda}\cdot\mathbf{Q}(t) - 2Z\cdot\mathbf{Q}(t)\\ &\leq BT - T\epsilon_g/2\sum_{m,u} Q_{m,u}(t),\end{aligned}$$

which gives us the required result.

Now to prove (C.9), Fix constants $\delta_1 \in (0,1)$ and $T \in \mathbb{N}$ such that

$$\delta_1 \leq \frac{\epsilon_g}{\bar{\mathsf{A}} + \epsilon_g/2},$$

and

$$\frac{2 + T\delta_1}{\epsilon_s} \leq \frac{T\epsilon_g}{2\bar{\mathsf{R}}}.$$

Define the random time Γ as

$$\Gamma := \max\left\{T_1(f(\delta_1/2)), \min\left\{t : \epsilon_l(t) \leq \delta_1/4\right\}\right\}.$$

From Lemma C.4, we have that $\mathbb{E}\left[\Gamma^2 \mid \mathbf{J}(0), \mathbf{Q}(1)\right] < \infty$. For any $l \geq 0$, $E_l(t+l)$ is independent of $Y(t), \mathcal{E}^0$ and if $E_l(t+l) = 0$, then $H(t+l)$ is independent of these random variables and distributed according to $\boldsymbol{\mu}$. This gives us

$$\begin{aligned}
&\mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t+l),H(t+l))} \alpha^*_{\mathbf{r}}(\mathbf{J}(t+l), H(t+l))\mathbf{r} \;\middle|\; Y(t), \mathcal{E}^0\right] \\
&\geq \sum_{l=0}^{T-1}\Bigg(\mathbb{P}\left[E_l(t+l)=0\right]\times \\
&\quad \sum_{j\in\mathcal{J}} \mathbb{P}\left[\mathbf{J}(t+l)=j \;\middle|\; E_l(t+l)=0, Y(t), \mathcal{E}^0\right]\mathbb{E}\left[\sum_{\mathbf{r}\in\mathcal{R}(j,H(t+l))} \alpha^*_{\mathbf{r}}(j, H(t+l))\mathbf{r}\right]\Bigg) \\
&= \sum_{l=0}^{T-1}\Bigg((1-\epsilon_l(t+l))\times \\
&\quad \sum_{j\in\mathcal{J}} \mathbb{P}\left[\mathbf{J}(t+l)=j \;\middle|\; E_l(t+l)=0, Y(t), \mathcal{E}^0\right]\left(\sum_{h\in\mathcal{H}}\mu(h)\sum_{\mathbf{r}\in\mathcal{R}(j,h)} \alpha^*_{\mathbf{r}}(j,h)\mathbf{r}\right)\Bigg) \\
&\geq (1-\delta_1/4)\sum_{l=0}^{T-1}\left(\left(\sum_{j\in\mathcal{J}}\sigma^*_j\sum_{h\in\mathcal{H}}\mu(h)\sum_{\mathbf{r}\in\mathcal{R}(j,h)} \alpha^*_{\mathbf{r}}(j,h)\mathbf{r}\right) - \|\mathbf{y}(t+l)-\boldsymbol{\sigma}^*\|_1\bar{\mathsf{R}}\right),
\end{aligned} \tag{C.12}$$

where for any $l \in [0, T-1]$,

$$y(t+l)_j = \mathbb{P}\left[\mathbf{J}(t+l)=j \;\middle|\; E_l(t+l)=0, Y(t), \mathcal{E}^0\right].$$

From Lemma C.5, given $\mathcal{E}^0$, for any $t > \Gamma$ and $l \in [0, T-1]$, we have

$$\|\mathbf{P}(t+l) - \mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)\|_1 \leq \delta_1.$$

Recall that

$$\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s) = \epsilon_s \mathbf{1}_{|\mathcal{J}|}\boldsymbol{\sigma}^* + (1-\epsilon_s)\mathbf{I}_{|\mathcal{J}|},$$

and for any $l \in \mathbb{N}$,

$$\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)^l = \left(1-(1-\epsilon_s)^l\right)\mathbf{1}_{|\mathcal{J}|}\boldsymbol{\sigma}^* + (1-\epsilon_s)^l\mathbf{I}_{|\mathcal{J}|}.$$

Since $\tau_1(\mathbf{1}_{|\mathcal{J}|}\boldsymbol{\sigma}^*) = 0$, using the definition in (4.10), it can be verified that $\tau_1(\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)^l) = (1-\epsilon_s)^l$. Therefore,

$$\Upsilon(\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)) = \sum_{l=0}^{\infty} \tau_1(\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)^l) = \frac{1}{\epsilon_s}.$$

From Lemma 4.3(a), we have

$$\begin{aligned}
\sum_{l=0}^{T-1} \|\mathbf{y}(t+l) - \boldsymbol{\sigma}^*\|_1 &\leq \sum_{l=0}^{T-1} \left(2\tau_1(\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s)^l) + \delta_1 \Upsilon(\mathbf{P}(\boldsymbol{\sigma}^*, \epsilon_s))\right) \\
&\leq \frac{2 + T\delta_1}{\epsilon_s} \\
&\leq \frac{T\epsilon_g}{2\bar{\mathsf{R}}}.
\end{aligned}$$

Since any solution to the linear program $L_{\mathbf{c}^{\epsilon_p}}(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)$ satisfies its constraints, we also have

$$\sum_{j \in \mathcal{J}} \sigma^*(\boldsymbol{\mu}, \boldsymbol{\lambda} + \epsilon_g)_j \sum_{h \in \mathcal{H}} \mu(h) \sum_{\mathbf{r} \in \mathcal{R}(j,h)} \alpha^*_{\mathbf{r}}(j,h)\mathbf{r} \geq \boldsymbol{\lambda} + \epsilon_g.$$

Using this in (C.11) and (C.10) gives us

$$\begin{aligned}
Z &\geq T(\boldsymbol{\lambda} + \epsilon_g/2)(1 - \delta_1/4) \\
&\geq T(\boldsymbol{\lambda} + \epsilon_g/2)\left(1 - \frac{\epsilon_g/4}{\bar{\mathsf{A}} + \epsilon_g/2}\right) \\
&\geq T(\boldsymbol{\lambda} + \epsilon_g/4),
\end{aligned}$$

and this proves (C.9). □

Combining Lemmas C.2 and C.3, we have part 2 of Theorem 4.3.

Bibliography

[1] Ftc policy statement on deception, October 1983.

[2] Self-regulatory principles for online behavioral advertising, July 2009.

[3] Major marketing / media trade groups launch program to give consumers enhanced control over collection and use of web viewing data for online behavioral advertising, October 2010.

[4] .com disclosures: How to make effective disclosures in digital advertising, March 2013.

[5] Iab native advertising playbook, December 2013.

[6] Native advertising roundup, October 2014.

[7] State of native advertising 2014, 2014.

[8] Ali Abbasi and Majid Ghaderi. Distributed base station activation for energy-efficient operation of cellular networks. In *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*, pages 427–436. ACM, 2013.

[9] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[10] John Abraham. The pharmaceutical industry as a political player. *The Lancet*, 360(9344):1498–1502, 2002.

[11] Deepak Agarwal. Detecting anomalies in cross-classified streams: a bayesian approach. *Knowledge and information systems*, 11(1):29–44, 2007.

[12] Charu C Aggarwal and Philip S Yu. Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM, 2001.

[13] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, June 2012.

[14] Reed Albergotti. Facebook to clean up news feeds. *The Wall Street Journal*, 2014.

[15] M. Andrews, K. Kumaran, K. Ramanan, A.L. Stolyar, R. Vijayakumar, and P. Whiting. CDMA data QoS scheduling on the forward link with variable channel conditions. *Bell Labs Tech. Memo*, April 2000.

[16] Jac M Anthonisse and Henk Tijms. Exponential convergence of products of stochastic matrices. *Journal of Mathematical Analysis and Applications*, 59(2):360–364, 1977.

[17] Oliver Arnold, Fred Richter, Gerhard Fettweis, and Oliver Blume. Power consumption modeling of different base station types in heterogeneous cellular networks. In *2010 Future Network & Mobile Summit*. IEEE, 2010.

[18] Karl J Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[19] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on Learning Theory-2010*, pages 13–p, 2010.

[20] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[21] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[22] Daniel Barbara, Ningning Wu, and Sushil Jajodia. Detecting novel network intrusions using bayes estimators. In *SDM*, pages 1–17. SIAM, 2001.

[23] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[24] Joeran Beel, Stefan Langer, and Marcel Genzmehr. Sponsored vs. organic (research paper) recommendations and the impact of labeling. In *Research and Advanced Technology for Digital Libraries*, pages 391–395. Springer, 2013.

[25] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[26] Dirk Bergemann and Alessandro Bonatti. Targeting in advertising markets: implications for offline versus online media. *The RAND Journal of Economics*, 42(3):417–443, 2011.

[27] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic Programming*. Anthropological Field Studies. Athena Scientific, 1996.

[28] Naga Bhushan, Junyi Li, Durga Malladi, Rob Gilmore, Dean Brenner, and Aleksandar Damnjanovic. Network densification: the dominant theme for wireless evolution into 5G. *IEEE Communications Magazine*, 52(2):82–89, 2014.

[29] David Blumenthal. Doctors and drug companies. *New England Journal of Medicine*, 351(18):1885–1890, 2004.

[30] Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical Science*, pages 235–249, 2002.

[31] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[32] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Machine Learning*, 5(1):1–122, 2012.

[33] Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Bounded regret in stochastic multi-armed bandits. *arXiv preprint arXiv:1302.1611*, 2013.

[34] Robin Burke, Bamshad Mobasher, Chad Williams, and Runa Bhaumik. Classification features for attack detection in collaborative recommender systems. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–547. ACM, 2006.

[35] C Buyukkoc, P Varaiya, and J Walrand. The cμ rule revisited. *Advances in applied probability*, 17(1):237–238, 1985.

[36] Nicolò Cesa-Bianchi and Paul Fischer. Finite-time regret bounds for the multiarmed bandit problem. In *ICML*, pages 100–108. Citeseer, 1998.

[37] Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

[38] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[39] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

[40] P. Chaporkar and S. Sarkar. Stable scheduling policies for maximizing throughput in generalized constrained queueing. In *IEEE Infocom*, 2006.

[41] Richard Combes, Chong Jiang, and Rayadurgam Srikant. Bandits with budgets: Regret lower bounds and optimal algorithms. In *Proceedings of the 2015*

ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pages 245–257. ACM, 2015.

[42] DR Cox and WL Smith. Queues. *Wiley*, 1961.

[43] Aidan Crook and Sanaz Ahari. Personalized search for everyone. bing blog, 2011.

[44] MR Davidson. Stability of the extreme point set of a polyhedron. *Journal of optimization theory and applications*, 90(2):357–380, 1996.

[45] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. A large-scale evaluation and analysis of personalized search strategies. In *Proceedings of the 16th international conference on World Wide Web*, pages 581–590. ACM, 2007.

[46] The Lancet Editorial. The uk drug industry: responsible, ethical, and professional? *The Lancet*, 366(9500):1828, 2005.

[47] Salah-Eddine Elayoubi, Louai Saker, and Tijani Chahed. Optimal control for base station sleep mode in energy efficient radio access networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 106–110. IEEE, 2011.

[48] Dag Elgesem. Search engines and the public use of reason. *Ethics and information Technology*, 10(4):233–242, 2008.

[49] Robert M Entman. Framing bias: Media in the distribution of power. *Journal of communication*, 57(1):163–173, 2007.

[50] Robert Epstein. How google could end democracy. *U.S. News & World Report*, 2014.

[51] Atilla Eryilmaz and R Srikant. Joint congestion control, routing, and mac for stability and fairness in wireless networks. *Selected Areas in Communications, IEEE Journal on*, 24(8):1514–1524, 2006.

[52] Adriane Fugh-Berman and Shahram Ahari. Following the script: how drug reps make friends and influence doctors. *PLoS Medicine*, 4(4):e150, 2007.

[53] Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations. *IEEE/ACM Transactions on Networking (TON)*, 20(5):1466–1478, 2012.

[54] Pedro Galeano, Daniel Peña, and Ruey S Tsay. Outlier detection in multivariate time series by projection pursuit. *Journal of the American Statistical Association*, 101(474):654–669, 2006.

[55] Aurélien Garivier and Olivier Cappé. The kl-ucb algorithm for bounded stochastic bandits and beyond. *arXiv preprint arXiv:1102.2490*, 2011.

[56] L. Georgiadis, M. J. Neely, and L. Tassiulas. *Resource Allocation and Cross-Layer Control in Wireless Networks.* NOW Publishers, Foundations and Trends in Networking, 2006.

[57] Anindya Ghose and Sha Yang. An empirical analysis of search engine advertising: Sponsored search in electronic markets. *Management Science*,

55(10):1605–1622, 2009.

[58] John C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

[59] Ben Goldacre. *Bad Pharma: How drug companies mislead doctors and harm patients*. Macmillan, 2014.

[60] Jie Gong, John S Thompson, Sheng Zhou, and Zhisheng Niu. Base station sleeping and resource allocation in renewable energy powered cellular networks. *IEEE Trans. on Communications*, 62(11):3801–3813, 2014.

[61] Aditya Gopalan, Constantine Caramanis, and Sanjay Shakkottai. On wireless scheduling with partial channel-state information. *Information Theory, IEEE Transactions on*, 58(1):403–420, 2012.

[62] Laura A Granka. The politics of search: A decade retrospective. *The Information Society*, 26(5):364–374, 2010.

[63] Xueying Guo, Zhisheng Niu, Sheng Zhou, and PR Kumar. Delay-constrained energy-optimal base station sleeping control. *IEEE Journal on Selected Areas in Communications*, 34(5):1073–1085, 2016.

[64] Deepali Gupta. Cci charges google with rigging search results; flipkart, facebook corroborate complaints. *The Economic Times*, 2015.

[65] Feng Han, Zoltan Safar, W Sabrina Lin, Yan Chen, and KJ Ray Liu. Energy-efficient cellular network operation via base station cooperation. In *2012*

IEEE International Conference on Communications (ICC), pages 4374–4378. IEEE, 2012.

[66] Ziaul Hasan, Hamidreza Boostanimehr, and Vijay K Bhargava. Green cellular networks: A survey, some research issues and challenges. *IEEE Communications surveys & tutorials*, 13(4):524–540, 2011.

[67] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *Data warehousing and knowledge discovery*, pages 170–180. Springer, 2002.

[68] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650, 2003.

[69] Bryan Horling and Matthew Kulick. Personalized search for everyone. Google Official Blog, 2009.

[70] Zan Huang, Daniel D Zeng, and Hsinchun Chen. Analyzing consumer-product graphs: Empirical findings and applications in recommender systems. *Management science*, 53(7):1146–1164, 2007.

[71] Lucas D Introna and Helen Nissenbaum. Shaping the web: Why the politics of search engines matters. *The information society*, 16(3):169–185, 2000.

[72] Peter Jacko. Restless bandits approach to the job scheduling problem and its extensions. *Modern trends in controlled stochastic processes: theory and applications*, pages 248–267, 2010.

[73] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. Neuro-fuzzy and soft computing, a computational approach to learning and machine intelligence. 1997.

[74] Gong Jie, Zhou Sheng, and Niu Zhisheng. A dynamic programming approach for base station sleeping in cellular networks. *IEICE transactions on communications*, 95(2):551–562, 2012.

[75] Ioannis Kamitsos, Lachlan Andrew, Hongseok Kim, and Mung Chiang. Optimal sleep patterns for serving delay-tolerant jobs. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, pages 31–40. ACM, 2010.

[76] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[77] Frank P Kelly, Aman K Maulloo, and David KH Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, pages 237–252, 1998.

[78] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *Information Theory, IEEE Transactions on*, 56(6):2980–2998, 2010.

[79] Roman Khazankin, Harald Psaier, Daniel Schall, and Schahram Dustdar. Qos-based task scheduling in crowdsourcing environments. In *Service-Oriented Computing*, pages 297–311. Springer, 2011.

[80] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[81] S. Krishnasamy, P. T. Akhil, A. Arapostathis, S. Shakkottai, and R. Sundaresan. Augmenting max-weight with explicit learning for wireless scheduling with switching costs. Tech. Report, UT Austin, January 2017.

[82] Subhashini Krishnasamy, Rajat Sen, Ramesh Johari, and Sanjay Shakkottai. Regret of queueing bandits. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1669–1677. 2016.

[83] Subhashini Krishnasamy, Rajat Sen, Sewoong Oh, and Sanjay Shakkottai. Detecting sponsored recommendations. In *The 2015 ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '15. ACM, 2015. Poster Paper.

[84] Subhashini Krishnasamy, Rajat Sen, Sewoong Oh, and Sanjay Shakkottai. Detecting sponsored recommendations. *arXiv preprint arXiv:1504.03713*, 2015.

[85] Subhashini Krishnasamy, Rajat Sen, Sanjay Shakkottai, and Sewoong Oh. Detecting sponsored recommendations. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 2(1):6:1–6:29, November 2016.

[86] Christopher Kruegel and Giovanni Vigna. Anomaly detection of web-based

attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, pages 251–261. ACM, 2003.

[87] Srisankar Kunniyur and Rayadurgam Srikant. End-to-end congestion control schemes: Utility functions, random losses and ecn marks. *Networking, IEEE/ACM Transactions on*, 11(5):689–702, 2003.

[88] Harold Kushner. *Heavy traffic analysis of controlled queueing and communication networks*, volume 47. Springer Science & Business Media, 2013.

[89] Khaled Labib and Rao Vemuri. Nsom: A real-time network-based intrusion detection system using self-organizing maps. *Networks and Security*, pages 1–6, 2002.

[90] Sébastien Lahaie, David M Pennock, Amin Saberi, and Rakesh V Vohra. Sponsored search auctions. *Algorithmic game theory*, pages 699–716, 2007.

[91] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[92] C Seth Landefeld and Michael A Steinman. The neurontin legacy–marketing through misinformation and manipulation. *New England Journal of Medicine*, 360(2):103, 2009.

[93] Steven H Landers and Ashwini R Sehgal. Health care lobbying in the united states. *The American journal of medicine*, 116(7):474–477, 2004.

[94] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*, pages 25–36. SIAM, 2003.

[95] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[96] X. Lin, N. Shroff, and R. Srikant. A tutorial on cross-layer optimization in wireless networks. *IEEE Journal on Selected Areas in Comm.*, 2006.

[97] Xiaojun Lin and Ness B Shroff. Joint rate control and scheduling in multihop wireless networks. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1484–1489. IEEE, 2004.

[98] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*, 2010.

[99] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[100] Christopher Lott and Demosthenis Teneketzis. On the optimality of an index rule in multichannel allocation for single-hop mobile networks with multiple

service classes. *Probability in the Engineering and Informational Sciences*, 14:259–297, 2000.

[101] Aditya Mahajan and Demosthenis Teneketzis. Multi-armed bandit problems. In *Foundations and Applications of Sensor Management*, pages 121–151. Springer, 2008.

[102] Matthew V Mahoney and Philip K Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 376–385. ACM, 2002.

[103] M Ajmone Marsan, Luca Chiaraviglio, Delia Ciullo, and Michela Meo. Optimal energy savings in cellular access networks. In *2009 IEEE International Conference on Communications Workshops*, pages 1–5. IEEE, 2009.

[104] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[105] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[106] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology (TOIT)*, 7(4):23, 2007.

[107] Michael J Neely, Eytan Modiano, and Chih-Ping Li. Fairness and optimal stochastic control for heterogeneous networks. *Networking, IEEE/ACM Transactions on*, 16(2):396–409, 2008.

[108] Michael J. Neely, Eytan Modiano, and Charles E. Rohrs. Tradeoffs in delay guarantees and computation complexity for n n packet switches. In *Proceedings of CISS*, 2002.

[109] José Niño-Mora. Marginal productivity index policies for scheduling a multiclass delay-/loss-sensitive queue. *Queueing Systems*, 54(4):281–312, 2006.

[110] José Niño-Mora. Dynamic priority allocation via restless bandit marginal productivity indices. *Top*, 15(2):161–198, 2007.

[111] Eunsung Oh, Bhaskar Krishnamachari, Xin Liu, and Zhisheng Niu. Toward dynamic energy-efficient operation of cellular network infrastructure. *IEEE Communications Magazine*, 49(6):56–61, 2011.

[112] Eunsung Oh, Kyuho Son, and Bhaskar Krishnamachari. Dynamic base station switching-on/off strategies for green cellular networks. *IEEE transactions on wireless communications*, 12(5):2126–2136, 2013.

[113] Spiros Papadimitriou, Hiroyuki Kitagawa, Philip B Gibbons, and Christos Faloutsos. Loci: Fast outlier detection using the local correlation integral. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 315–326. IEEE, 2003.

[114] Vianney Perchet, Philippe Rigollet, Sylvain Chassang, and Erik Snowberg. Batched bandit problems. *arXiv preprint arXiv:1505.00369*, 2015.

[115] Chad Pollitt. Everything you need to know about sponsored content. January 2015.

[116] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD Record*, volume 29, pages 427–438. ACM, 2000.

[117] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[118] Paat Rusmevichientong and David P Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 260–269. ACM, 2006.

[119] Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

[120] Antoine Salomon, Jean-Yves Audiber, and Issam El Alaoui. Lower bounds and selectivity of weak-consistent policies in stochastic multi-armed bandit problem. *The Journal of Machine Learning Research*, 14(1):187–207, 2013.

[121] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[122] Steven L Scott. A modern bayesian look at the multi-armed bandit. *Appl. Stoch. Models in Business and Industry*, 26(6):639–658, 2010.

[123] E Seneta. Sensitivity of finite markov chains under perturbation. *Statistics & probability letters*, 17(2):163–168, 1993.

[124] Eugene Seneta. *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

[125] Anup Shah. Pharmaceutical corporations and medical research. *Global Issues*, 2010.

[126] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4):440–448, 2002.

[127] R. Srikant and L. Ying. *Communication Networks – An Optimization, Control, and Stochastic Networks Perspective*. Cambridge University Press, 2014.

[128] R. Srikant and Lei Ying. *Communication Networks: An Optimization, Control and Stochastic Networks Perspective*. Cambridge University Press, 2014.

[129] Alexander L Stolyar. Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm. *Queueing Systems*, 50(4):401–457, 2005.

[130] L. Tassiulas. Linear complexity algorithms for maximum throughput in radio networks and input queued switches. In *IEEE Infocom*, 1998.

[131] Herman Tavani. Search engines and ethics. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Spring 2014 edition, 2014.

[132] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, pages 285–294, 1933.

[133] Alexandre B Tsybakov. *Introduction to nonparametric estimation*. Springer Science & Business Media, 2008.

[134] Catherine Tucker. Social advertising. *SSRN eLibrary*, 2012.

[135] A.M. Turpin and J.B. Katz. System and method for implementing advertising in an online social network, August 7 2008. US Patent App. 12/011,880.

[136] Jan A Van Mieghem. Dynamic scheduling with convex delay costs: The generalized c mu rule. *The Annals of Applied Probability*, pages 809–833, 1995.

[137] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J Weinberger. Inequalities for the l1 deviation of the empirical distribution, 2003. HP Labs Technical Report.

[138] John T Wen and Murat Arcak. A unifying passivity framework for network flow control. *Automatic Control, IEEE Transactions on*, 49(2):162–174, 2004.

[139] R. J. B. Wets. On the continuity of the value of a linear program and of related polyhedral-valued multifunctions. *Mathematical programming study*, (24):14–29, 1985.

[140] Ward Whitt. Heavy traffic limit theorems for queues: a survey. In *Mathematical Methods in Queueing Theory*, pages 307–350. Springer, 1974.

[141] Gang Wu, Chenyang Yang, Shaoqian Li, and Geoffrey Ye Li. Recent advances in energy-efficient networks and their application in 5g systems. *IEEE Wireless Communications*, 22(2):145–151, 2015.

[142] Jingjin Wu, Yujing Zhang, Moshe Zukerman, and Edward Kai-Ning Yung. Energy-efficient base-stations sleep-mode techniques in green cellular networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(2):803–826, 2015.

[143] Sha Yang and Anindya Ghose. Analyzing the relationship between organic and sponsored search advertising: Positive, negative, or zero interdependence? *Marketing Science*, 29(4):602–623, 2010.

[144] Dit-Yan Yeung and Calvin Chow. Parzen-window network intrusion detectors. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 4, pages 385–388. IEEE, 2002.

[145] Yung Yi, Alexandre Proutiere, and Mung Chiang. Complexity in wireless scheduling: Impact and tradeoffs. In *Proc. of the 9th ACM International Symp. on Mobile Ad Hoc Networking and Computing (MobiHoc)*, 2008.

[146] Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.

[147] Jianchao Zheng, Yueming Cai, Xianfu Chen, Rongpeng Li, and Honggang Zhang. Optimal base station sleeping in green cellular networks: A distributed cooperative framework based on game theory. *IEEE Transactions on Wireless Communications*, 14(8):4391–4406, 2015.

[148] Michael Zimmer. The value implications of the practice of paid search. *Bulletin of the American Society for Information Science and Technology*, 32(2):23–25, 2006.

Vita

Subhashini Krishnasamy received her Master's degree in Telecommunications from the Indian Institute of Science (IISc), Bangalore and Bachelor's degree in Electronics and Communication Engineering from JNTU College of Engineering, Hyderabad. Prior to joining UT Austin, she was employed at Texas Instruments, Bangalore, where she worked on designing low-power WiFi transceivers. Her research focuses on designing learning and control algorithms for networks, especially of large scale.

Permanent address: subhashini.kb@utexas.edu

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.