Online Learning and Optimization in Operations Management

by

Rui Sun

B.S., Tsinghua University (2014)

S.M., Massachusetts Institute of Technology (2017)

Submitted to the Institute of Data, Systems, and Society
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Social Engineering Systems and Statistics
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2020

Author . . . .
Institute of Data, Systems, and Society
August 19 2020

Certified by . . . .
David Simchi-Levi
Professor of Engineering Systems
Thesis Supervisor

Certified by . . . .
Jinhua Zhao
Edward and Joyce Linde Associate Professor of City and
Transportation Planning
Thesis Committee Member

Certified by . . . .
Negin Golrezaei
Assistant Professor of Operations Management
Thesis Committee Member

Accepted by . . . .
Ali Jadbabaie
Director, Institute for Data, Systems, and Society; Program Chair

Online Learning and Optimization in Operations Management

by

Rui Sun

Submitted to the Institute of Data, Systems, and Society
on August 19 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Social Engineering Systems and Statistics

Abstract

We study in this thesis online learning and optimization problems in operations management where we need to make decisions in the face of incomplete information and operational constraints in a dynamic environment.

We first consider an online matching problem where a central platform needs to match a number of limited resources to different groups of users that arrive sequentially over time. The platform does not know the reward of each matching option and must learn the true rewards from the matching results. We formulate the problem as a Markovian multi-armed bandit with budget constraints, and propose an innovative algorithm that is based on assembling the policies for each single arm. We prove the algorithm's worst-case performance guarantee, and numerically show the algorithm's robust performance compared to alternative heuristics.

We next consider a revenue management problem with add-on discounts where a retailer offers discounts on selected supportive products (e.g. video games) to customers who have also purchased the core products (e.g. video game consoles). When the products' demand functions are unknown, we propose a UCB-based learning algorithm that uses the an FPTAS optimization algorithm as a subroutine to determine the prices of different types of products. We show that the algorithm can converge to the optimal full-information pricing policy. We also conduct numerical experiments with real-world data to illustrate the performance of our algorithm and the advantage of using the add-on discount strategy in practice.

We last consider a network revenue management problem where a retailer aims to maximize revenue from multiple products with limited inventory. The retailer does not know the demand of different products, and must learn demand from the sales data. To optimize the pricing decisions, we propose an efficient algorithm that combines the Thompson sampling technique and the online gradient descent method with a primal-dual framework. In comparison to traditional algorithms that are based on frequently solving linear programs, our algorithm does not need to solve any linear program, and therefore, has the advantage in computational efficiency. We analyze the performance guarantee of our algorithm, and show the algorithm's fast running time through numerical experiments.

Thesis Supervisor: David Simchi-Levi
Title: Professor of Engineering Systems

Thesis Committee Member: Jinhua Zhao
Title: Edward and Joyce Linde Associate Professor of City and Transportation Planning

Thesis Committee Member: Negin Golrezaei
Title: Assistant Professor of Operations Management

Acknowledgments

This thesis is a reflection of my rewarding and fulfilling journey at MIT, and I feel fortunate and privileged to meet so many incredible people along this wonderful journey.

First and foremost, I would like to thank my advisor, David Simchi-Levi, who has always been a great mentor. I worked with David first as a master student at CEE and later as a doctoral student at IDSS. During the past six years, I have received tremendous support from David, and all the things I have accomplished during my PhD would be impossible without his guidance and support. I always admire his sharp mind, his passion about work, his charisma and most importantly, his wisdom. I feel so grateful to have David as my advisor.

Next, I would like to thank my thesis committee members, Jinhua Zhao and Negin Golrazeai, for their time and effort. I took a class on behavior and policy from Jinhua when I was a master student at CEE. The class has broadened my horizons, and shown me so many interesting angles of viewing a research question. I was also fortunate to be a teaching assistant for Negin in her operations management class, where I got the opportunity to sharpen my skills, and learn about a wide range of applications of operations management in the real world. I have also learned a lot about teaching and managing a class of students from that wonderful experience.

I would also like to appreciate the support from IDSS. I would like to thank the directors, Munther Dahleh and Ali Jadbabaie for their great job creating a collaborative and intellectually exhilarating environment. I am also indebted to many IDSS staff, especially Janet Kerrigan, Elizabeth Miles. Moreover, I would like to thank a few other MIT faculties, Stephen Graves, Richard Larson and Vincent Chan, for their guidance and support.

This thesis is a result of collaboration with some of the greatest minds in our field. Chapter 2 is a joint work with Xinshang Wang during his time at the Data Science Lab. Chapter 3 is a joint work with Huanan Zhang during my internship at the Alibaba Group. Chapter 4 is also a joint work with Xinshang, and it is motivated by the work I have done as a research intern at Alibaba under the supervision of Prof. Wotao Yin. I have also received numerous help from Michelle and my fellow students in the Data Science Lab. I appreciate their time for the valuable discussions and their candid comments on improving my work. All of these people are talented researchers and good friends, and I am very fortunate to work with them.

I also want to say thank you to all my friends at MIT: Zhaoyuan, Tianyi, Heng, Qingying, Peter, Louis, Yiqun, Jinzhi, Hanzhang, Li, Ruihao, Hanwei, Jinglong, Xiaoyue, Yunzong and many others. I would never forget the wonderful times we spend together and the beautiful memories we share. Thank you all for making my PhD life amazing and colorful.

Finally, I owe my deepest gratitude to my parents who have been providing me with their endless support and unconditional love. This thesis is dedicated to them.

Contents

1 Introduction **15**

1.1 Motivation 15

1.2 Overview 17

2 Online Matching with Bayesian Rewards **21**

2.1 Introduction 21

2.1.1 Literature Review 25

2.2 Model Formulation 30

2.2.1 Online Matching Process 30

2.2.2 Bayesian Online Learning 31

2.2.3 Preliminaries 34

2.3 Algorithms for Two Special Cases 37

2.3.1 Single time period and multiple resources 38

2.3.2 Single resource and multiple time periods 39

2.4 Algorithm for the General Problem 47

2.5 Numerical Experiments 51

2.5.1 Alternative algorithms 51

2.5.2 Experiment setup 52

2.5.3 Experiment results and analysis 53

3 Online Learning and Optimization for Add-on Discounts **59**

3.1 Introduction 59

3.1.1 Literature Review 63

3.2 Model 66
3.3 Optimization Subroutine 69
3.4 Learning Algorithm and Regret Analysis 74
3.4.1 The learning algorithm 75
3.4.2 Regret analysis 76
3.5 Discussion 80
3.5.1 Assumption on independent demand 80
3.5.2 Assumption on add-on demand function 82
3.5.3 Assumption on Bernoulli demand 84
3.6 Numerical Experiments 84
3.6.1 Experiment settings 85
3.6.2 Result Analysis 86

4 Fast Thompson Sampling for Online Network Revenue Management 91
4.1 Introduction 91
4.1.1 Literature Review 94
4.2 Fast Thompson Sampling 97
4.2.1 Model 97
4.2.2 Relation to the Multi-Armed Bandit Problem 98
4.2.3 Algorithm 99
4.2.4 Comparison to LP-based Thompson Sampling 101
4.2.5 Comparison to primal-dual bandit with knapsack 102
4.2.6 Examples 103
4.3 Performance Analysis 104
4.3.1 Benchmark 105
4.3.2 Analysis of Fast Thompson Sampling 106
4.3.3 Numerical Experiments 109
4.4 Extensions and Further Applications 115
4.4.1 Contextual Model 115
4.4.2 Linear Contextual Model 118

5 Concluding Remarks **123**

5.1 Summary and Future Research Directions 123

A Technical Results in Chapter 2 **129**

A.1 Proof of Theorem 2.1 129

A.2 Proof of Theorem 2.2 131

A.3 Proof of Theorem 2.3 133

A.4 Proof of Theorem 2.4 133

A.5 Proof of Proposition 2.1 135

A.6 Proof of Proposition 2.2 136

A.7 Proof of Lemma 2.1 137

A.8 Proof of Lemma 2.2 138

A.9 Proof of Lemma A.1 139

B Technical Results in Chapter 3 **141**

B.1 Proof of Lemma 3.1 141

B.2 Proof of Lemma 3.3 142

B.3 Proof of Lemma 3.4 143

B.4 Proof of Lemma 3.5 143

B.5 Proof of Lemma 3.6 145

C Technical Results in Chapter 4 **149**

C.1 Efficient Projection Algorithms 149

C.2 Primal-dual Decomposition 149

C.3 Proof of Proposition 4.1 151

C.4 Proof of Proposition 4.2 155

C.5 Proof of Proposition 4.3 157

C.6 Proof of Theorem 4.3 159

List of Figures

2-1 A canonical matching problem for online traffic allocation platforms. . 22
2-2 A pictorial explanation of the lower bound of an algorithm's expected reward under Framework. 41
2-3 Sets of arms divided by the cutting point: G_0(I), G_1(II), G_2(III). . . . 43
2-4 A pictorial explanation of the performance guarantee of algorithm Prophet. 46
2-5 Function curves of $F_i(x)$ for $i \in [N]$ in Setting 1 57
2-6 Function curves of $F_i(x)$ for $i \in [N]$ in Setting 2 57

3-1 Add-on discount example from Gamestop Corp, retrieved on 2019-08-18. 60
3-2 Example of price inconsistency for the same selection of products. Retrieved on 2019-10-23 from Amazon.com. 61
3-3 Illustration of a customer's purchase dynamics. 67
3-4 Performance of algorithm UCB-Add-On in the test case for $S = 6$ with medium add-on discount effect. 89

4-1 Performance comparison of different algorithms: single product example with $I = 0.25T$. 110
4-2 Performance comparison of different algorithms: single product example with $I = 0.5T$. 111
4-3 Performance comparison of different algorithms: multiple product linear demand with $I = 3T$. 113
4-4 Performance comparison of different algorithms: multiple product linear demand with $I = 15T$. 113

4-5 Performance comparison of different algorithms: multiple product exponential demand with $I = 5T$. 114

4-6 Performance comparison of different algorithms: multiple product exponential demand with $I = 12T$. 114

4-7 Performance comparison of different algorithms: multiple product logit demand with $I = 7T$. 115

4-8 Performance comparison of different algorithms: multiple product logit demand with $I = 30T$. 115

List of Tables

2.1 Comparison of algorithms under Setting 1 53

2.2 Comparison of algorithms under Setting 2 54

2.3 Average number of pulls from each period in Setting 1 56

2.4 Average number of pulls from each resource in Setting 1 56

2.5 Average number of pulls from each time period in Setting 2 57

2.6 Average number of pulls from each resource in Setting 2 57

3.1 Summary of experiment results under different test scenarios. 88

Chapter 1

Introduction

1.1 Motivation

Operations Management is concerned with the decision-making process that aims to increase the efficiency of business operations and the production of goods or services. With the increase of data availability and advance of information technologies, modern operations management has been increasingly relied on data-driven decision-making systems, and one of the key challenges in designing such systems is to develop good operational policies in the face of uncertainty. For example, when an online retailer determines the prices of products, she faces the uncertainty of customer demand under different prices; when an online ad allocation platform decides the selection of ads shown to each user, the platform faces the uncertainty of the user's click response towards different ads. The omnipresence of uncertainties in these applications motivates the study of algorithms that can learn through interactions with the environment and optimize operational decisions in an online fashion.

When developing online algorithms for operations management problems under uncertainty, one faces the classic trade-off between exploration and exploitation. Here, exploration describes the process of trying different actions to gain more knowledge about the uncertain factors so as to improve future decisions, and exploitation describes the process of taking the action that optimizes the system based on the knowledge obtained so far. In practice, the total planning horizon is usually limited. If a policy spends too many time periods on exploration, the time left for exploitation will be very limited, and if a policy uses very few periods for exploration, the estimation accuracy of the unknown factors will be low and affect the decisions in the exploitation phase. In both scenarios, the policy cannot achieve a good performance.

One classic tool to study this exploration-exploitation trade-off is the multi-armed bandit model. Specifically, in a multi-armed bandit problem, each action is modeled as an arm, and every time an agent pulls the arm, the agent observes a random reward that follows certain probability distribution. The agent does not know the reward distribution of each arm, and therefore, needs to learn the knowledge by sequentially experimenting different arms. The agent's goal is to maximize the total expected reward.

The multi-armed bandit model and its many variants have been studied extensively in the literature over the past decade, and they have been applied to diverse domains ranging from medical trials, to communication networks, to online advertising and revenue management. For example, in clinical trials, we can model experimental treatments as "arms" and use bandit algorithms to investigate the effects of different treatments with the goal of minimizing patient losses; in routing, we can model route choices as "arms" and apply bandit algorithms to adaptively optimize the routing solution in order to minimize delays in a network; and in online advertising, we can model selections of ads as "arms" and use bandit algorithms to find the optimal ad allocations to different types of web users so that the total number of ad clicks is maximized.

The growing interest to use multi-armed bandit techniques to solve online decision-making problems under uncertainty has also been seen in the Operation Management literature, where researchers focus on studying dynamic pricing, inventory control and assortment optimization problems under unknown demand or unknown choice models. Simultaneously, the operational constraints and practical restrictions one faces in solving these real-world problems, such as constraints on resource inventory and restrictions on switching decisions, have also reversely motivated new research directions for bandit algorithms and have made a big impact on the development of the bandit literature.

In this thesis, we study several operations management problems with a uncertain decision-making environment. We adopt the learning techniques from multi-armed bandit models to address the exploration-exploitation trade-offs in these problems. More importantly, we focus on incorporating the learning tasks into more complicated optimization systems to solve real-world problems that take into account practical constraints and operational limitations.

1.2 Overview

An overview of the works in this thesis is as follows.

In the first work, we study an online matching problem where a central platform needs to match a number of limited resources to different groups of users that arrive sequentially over time. The reward of each matching option depends both on the type of resource and the time period the user arrives. The matching rewards are assumed to be unknown, but drawn from probability distributions that are known *a priori*. The platform then needs to learn the true rewards online based on real-time observations of the matching results. The goal of the central platform is to maximize the total reward from all the matching results without violating the resource capacity constraints.

We formulate the matching problem with Bayesian rewards as a Markovian multi-armed bandit with budget constraints, where each arm corresponds to a pair of a resource and a time period. We devise our algorithm by first finding policies for each single arm separately via a relaxed linear program, and then "assembling" these policies together through judicious selection criteria and well-designed pulling orders. We prove that the expected reward of our algorithm is at least $(\sqrt{2}-1)/2$ of the expected reward of the optimal algorithm. In particular, in the single-resource case, we prove the ratio is at least $\sqrt{2}-1$. We also design numerical experiments to verify our algorithm's performance guarantee, and compare the algorithm with alternative heuristics to illustrate the algorithm's good and robust performance in various settings.

In the second work, we study a revenue management problem with add-on discounts. The problem is motivated by the practice in the video game industry, where a retailer offers discounts on selected *supportive* products (e.g. video games) to customers who have also purchased the *core* products (e.g. video game consoles). We formulate this problem as an optimization problem to determine the prices of different products and the selection of products with add-on discounts. To address the computational challenge of this optimization problem, we propose an efficient FPTAS algorithm that can solve the problem approximately to any desired accuracy.

Additionally, we consider the revenue management problem in the setting where the retailer has no prior knowledge of the demand functions of different products. To solve this problem, we propose a UCB-based learning algorithm that uses the FPTAS optimization algorithm as a subroutine. We show that our learning algorithm can converge to the optimal algorithm that has full information of the true demand functions, and we prove that the convergence rate is tight up to a logarithmic term.

We conduct numerical experiments with the real-world transaction data we collect from a popular video gaming brand's online store on Tmall.com. The numerical results illustrate our learning algorithm's robust performance and fast convergence to the optimal algorithm in various scenarios. Moreover, the results show that our learning algorithm can outperform the optimal policy that does not use any add-on discounts, which illustrates the advantages of using the add-on discount strategy in practice.

In the third work, we consider a canonical price-based network revenue management problem where a retailer with the goal of maximizing revenue needs to decide the prices of multiple products with limited inventory over a finite selling season. The demand of different products, as functions of prices, contain parameters that are unknown, and therefore, the retailer needs to learn these parameters from the sales data. In addition, we assume that the demand parameters are drawn from certain probability distributions that are known as prior knowledge.

In the presence of inventory constraints, the retailer faces the trade-off between exploring different prices to learn demand and exploiting the price that maximizes revenue based on the current estimations. To tackle this challenge, we propose in this work a novel primal-dual algorithm that uses the Thompson sampling algorithm to learn the unknown demand parameters, and uses the online gradient descent algorithm to learn the unit value of inventory. Compared to traditional algorithms that are based on solving linear programs (LP) to optimize price in each selling period, our algorithm does not solve any LP, and therefore, has the advantage in computational efficiency.

In this work, we provide the optimality performance guarantee of our primal-dual algorithm, which illustrates the algorithm's convergence to the optimal pricing algorithm that knows the true demand parameter. We also show by numerical experiments that our algorithm has an outstanding performance in both optimality and computation time in comparison to other algorithms in the literature. Moreover, we discuss extensions of our primal-dual algorithm framework, and show that the framework can be conveniently adapted to a variety of different problem settings.

The remainder of this thesis is organized as follows. In Chapter 2, we consider the online matching problem with Bayesian rewards. We formulate the problem as a Markovian multi-armed bandit with budget constraints and pulling order restrictions. We develop an innovative algorithm that is based on the idea of "assembling" single-arm policies, and provide the algorithm's worst-case performance guarantee.

In Chapter 3, we consider the add-on discount problem with unknown demand functions. We first develop an efficient FPTAS approximation algorithm to solve the problem under full demand information, and then develop a UCB-based learning algorithm that uses the FPTAS optimization algorithm as a subroutine to solve the problem under unknown demand. We also conduct numerical experiment based on real-world data to show the advantage of using the add-on discount strategy in practice.

In Chapter 4, we consider the online network revenue management problem with unknown demand parameters. We develop a novel primal-dual algorithm that uses Thompson sampling to learn demand and uses online gradient descent to learn the unit value of product inventory. We numerically show that our algorithm has the advantage in computational efficiency compared to other LP-based dynamic pricing and learning algorithms in the literature. We also provide the performance guarantee of our algorithm through an intuitive primal-dual analysis framework, and discuss extensions of our algorithm to a variety of dynamic pricing settings.

Finally, in Chapter 5, we conclude the thesis with discussions on further research directions and extensions based on the problems and algorithms we study in each chapter. The technical proofs for each chapter are included in the appendices.

Chapter 2

Online Matching with Bayesian Rewards

2.1 Introduction

In many real-world resource allocation problems, such as online ad allocation (see Mehta Panigrahi (2012), Mehta et al. (2013)) and appointment scheduling (see Truong (2015)), the central task is to effectively match a number of resources with limited capacities to heterogeneous customers that arrive sequentially over time, with the goal of maximizing a certain notion of total reward (e.g. total revenue or customer utility) from all the matching results.

For a concrete example, consider the following online traffic allocation problem faced by the central platform of a large e-commerce company. When a user visits the e-market, the platform may display a banner ad on the mobile app or the web page. Such ads come from different channels such as a live web-streaming platform, a coupon delivery system, or a product recommendation system, etc. Each ad is linked to a specific item that the channel itself selects for recommendation. The central platform needs to route online traffic (i.e., impressions) to all of these channels. In this traffic allocation scheme, each channel provides an estimate of the number of impressions it would like to receive. The central platform then needs to globally maximize the total number of clicks on the displayed ads from all the channels.

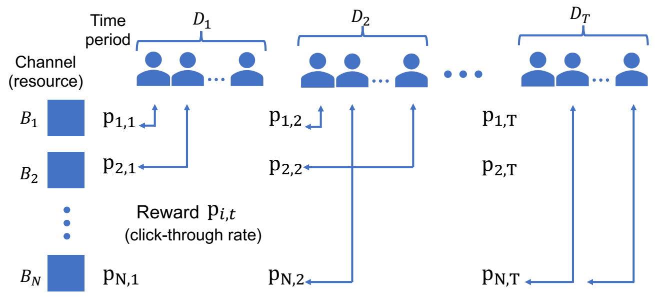


Figure 2-1: A canonical matching problem for online traffic allocation platforms.

Figure 2-1 illustrates the dynamics of a canonical matching problem faced by such an online traffic allocation platform. There are a number of different recommendation channels. Each channel corresponds to a resource associated with a "budget" value. The budget represents the number of impressions that the channel expects to receive. The central platform may not satisfy these expectations from all the channels (i.e., may not use up all the budget). Instead, it tries to meet the expectations by maximizing the total reward collected from the budget of these channels.

The total time horizon is divided into several discrete time periods, and in each time period, a group of users arrive at the central platform sequentially. When a user arrives, the platform needs to match the user to one of the channels. The reward of each matching option is measured by the user's click probability for that channel, which is also known as the channel's click-through rate. These click-through rates depend not only on the recommendation channel, but also the time period the user arrives. For instance, the click-through rate of a live web-streaming channel in the morning could be dramatically different from that in the evening. In addition, in this research, we do not consider the central platform's ability to estimate *personalized* click-through rates, as in practice some channels require the composition of the flow of impressions to remain unchanged. Nevertheless, the central platform has the ability to adjust *over time* the size of the impression flow that each channel receives.

Suppose the central platform knew the click-through rates of all the channels in all time periods. In order to maximize the total number of clicks, the platform needs to solve a global optimization problem to make allocation decisions. A common reward-maximization strategy is to solve a relaxed linear program

$$\begin{aligned}
\max_{\mathbf{x}} & \sum_{i,j} p_{i,t} x_{i,t} \\
\text{s.t.} & \sum_{t} x_{i,t} \leq B_i, \quad \forall i \\
& \sum_{i} x_{i,t} \leq D_t, \quad \forall t \\
& \mathbf{x} \geq 0,
\end{aligned}$$

where $p_{i,t}$ is the click-through rate for channel i in period t, B_i is the total budget of channel i, and D_t is the number of users in period t. Then the central platform would allocate approximately $x_{i,t}$ users to channel i in period t.

Unfortunately, in practice, it is usually a challenging task for the central platform to accurately estimate these click-through rates because of many *real-world* complications. First, for online platforms, click-through rates are often estimated by deep learning models. The estimation results are refreshed from time to time (depending on the availability of computational resources), which causes drastic changes in the predictions of future click-through rates. Second, different user types may arrive in different time periods, so a channel's click-through rate during the day can be very different from that during the evening. Due to such variations, the estimation of a channel's click-through rate in a certain time period may not be useful for future time periods.

Motivated by these challenges in estimating click-through rates, we study in this paper an online matching problem with unknown rewards. Specifically, the rewards of matching options, as shown in Figure 2-1, are not known *a priori*. In this setting, the central platform needs to decide which type of resource to allocate to each arriving user, while learning the reward of each matching option on the fly based on real-time observations of the matching results. Given the limited number of allocations both from each resource and in each time period, the central platform faces the trade-off between "exploiting" the matching options with demonstrably high rewards and "exploring" the options with uncertain reward estimations. More importantly, as future click-through rates may change according to the way each channel updates its recommendation strategy (e.g., applies a new dataset for training deep learning models), and because user types vary with time periods, the rewards of allocating the same resource in different time periods can be very different. This fact raises an additional challenge that the knowledge about the rewards of matching options the platform learns from one time period can hardly be applied to the next.

Our model features a Bayesian learning setting. That is, we assume the rewards of matching options, although unknown, are drawn randomly from certain known prior probability distributions. With such prior information, the central platform can make preliminary decisions regarding the allocation of each resource's budget across different time periods. The platform can then adjust these decisions adaptively as it gains more information about the true rewards online.

The reasons for adopting a Bayesian learning setting in our problem are as follows. First, in practice, the central platform can always obtain such prior knowledge of a channel's click-through rate by analyzing the channel's click history or from other similar channels. Second, for our matching problem, a Bayesian setting is less conservative than a non-Bayesian setting where no prior information is provided. In the latter setting, all click-through rates are totally unknown. Ball Queyranne (2009) and Ma Simchi-Levi (2019) have shown that in such a non-Bayesian environment, it might be difficult to design an online matching algorithm with a strong performance guarantee. [1]

The main contribution of this paper is that we propose an algorithm with a strong performance guarantee for the matching problem with Bayesian rewards. In particular, the optimal algorithm for the problem can be found via a dynamic program. However, such an approach suffers from *the curse of dimensionality*, and is thus computationally infeasible. We show in this paper that our matching and learning algorithm can obtain at least a constant fraction of the expected reward of the optimal

[1]Here, an algorithm's performance is measured by competitive ratios. Ball Queyranne (2009) and Ma Simchi-Levi (2019) have shown that it is impossible to obtain a constant competitive ratio when there exists arbitrarily many reward levels for each single resource.

algorithm. We state our main theoretical result in Theorem 2.4. This main result is based on an interesting analysis of a special case of the model that we present in Section 2.3.2. Besides technical analysis, we also numerically compare our algorithm against several other heuristics that are motived by the algorithms in Guha Munagala (2013). The numerical results show that our algorithm obtains around twice the expected reward of the randomized heuristic in all test cases, and outperforms the benchmark heuristic in the worst-case scenarios. The comparison results demonstrate our algorithm's good and robust performance in various settings.

2.1.1 Literature Review

This paper is related to two main streams of research topics in the literature: online resource allocation and online learning.

Online Resource allocation

The online resource allocation literature considers the problem of matching resources with limited capacities to customers that arrive sequentially over time. A major focus of the research papers in this area is to develop optimal allocation algorithms under different assumptions of customer arrival sequences. One of the classic settings assumes stochastic arrivals, where the reward associated with each arriving customer is assumed to be drawn i.i.d. from a known distribution. For a brief overview of the celebrated results in this setting, see Feldman et al. (2009), Feldman et al. (2010), Jaillet Lu (2012) and Jaillet Lu (2013). In another classic setting, the customer arrival sequence is assumed to be a random permutation of a unknown sequence, and a summary of the main results in this setting can be found in Goel Mehta (2008), Devanur Hayes (2009), Mahdian Yan (2011) and Agrawal et al. (2014).

We highlight the online matching setting that assumes arbitrary arrivals. In this setting, the central platform has no predictability in the types of customers that will arrive in the future. In this line of research, Ball Queyranne (2009) study an online booking problem. That paper considers a single resource that generates different rewards when allocated to different customers. Mehta et al. (2005) and Buchbinder et al. (2007) study the Adwords problem. Golrezaei et al. (2014) study a personalized online assortment problem. These papers consider multiple resources, and each resource generates the same reward rate when allocated to different customers. Ma Simchi-Levi (2019) generalize these two scenarios and study the problem that involves multiple resources with each resource having multiple reward rates.

The performance of algorithms in the arbitrary arrival setting is usually measured by *competitive ratio*, which is defined as the ratio between the reward of an algorithm and the reward of the optimal algorithm that knows the arrival sequence in advance. In the problem that considers multiple reward rates, Ball Queyranne (2009) and Ma Simchi-Levi (2019) show that the best competitive ratio an algorithm can achieve is dependent on the values of rewards, which means no algorithms can obtain a constant competitive ratio for arbitrary arrival sequences.

A recent stream of the research has been focusing on incorporating online learning into resource allocation problems where the rewards for the allocation decisions are unknown and the central platform has to learn these rewards online. In the setting of stochastic arrivals, the problem is closely related to the "bandits with knapsacks" problem that studies stochastic bandits with resource constraints. The bandits with knapsacks model is introduced in Slivkins (2019), and then followed by a stream of work, such as Agrawal Devanur (2014a), Agrawal Devanur (2016) Agrawal et al. (2016b), in which the authors extend the model to allow more generalized forms of rewards and constraints, and Ferreira et al. (2018), in which the authors extend the model to the Bayesian setting and study a personalized dynamic pricing problem. In the setting of arbitrary arrivals, Cheung et al. (2018) study a general resource allocation problem with unknown rewards. That paper provides a powerful algorithm framework that integrates the inventory balancing techniques for online matching problems with a broad class of online learning algorithms.

Building on this recent stream of work, we study in this paper an online resource allocation model with unknown rewards. Specifically, our model considers non-stationary customer arrivals, which means the rewards of allocation decisions in different time periods are different. This setting is less conservative than that in Ma Simchi-Levi (2019) and Cheung et al. (2018), and more flexible than that in Ferreira et al. (2018). Both our model and Ferreira et al. (2018) adopt a Bayesian learning setting. The difference is that Ferreira et al. (2018) provides a sub-linear regret as the algorithm's performance measure, which makes sense in the asymptotic regime as the number of time periods or users grows large. In this paper, we aim to develop an algorithm with a performance guarantee for all problem instances. Therefore, instead of providing a sub-linear regret, we show that our algorithm can obtain a constant fraction of the optimal algorithm's reward.

Online learning

The online learning literature focuses on sequential decision-making problems under unknown rewards, and the general approach is to formulate the learning (exploration) and earning (exploitation) trade-off in the decision-making process as a multi-armed bandit problem. Specifically in the Bayesian learning setting, the problem is usually formulated as a Markovian multi-armed bandit, where each arm is associated with a Markov decision process (MDP).

In Markovian bandit problems, the rewards and transition probabilities at each state of the MDPs are given as input, and therefore, the major challenge is computational. In the infinite-horizon setting with discounted rewards, the problem can be solved by the celebrated Gittins index policy. The books Gittins Jones (1979) and Gittins et al. (2011) have provided an in-depth treatment of such index-based policies.

Note that the Markovian bandit problem is different from the stochastic bandit problem. Specifically, the objective of the latter is to minimize regret, which is defined as an algorithm's reward loss relative to the optimal policy of a clairvoyant that knows the true rewards of each arm. For a comprehensive review on the stochastic bandit problem and its many variations, see Bubeck et al. (2012) and Slivkins (2019).

Building on the traditional Markvoian bandit model, a recent line of research has been focusing on studying the finite-horizon version of the model with extra operational constraints. In such cases, the optimal policy is computationally intractable, so an algorithm's performance is measured by the ratio between the expected reward of the algorithm and the expected reward of the optimal algorithm. For example, Farias Madan (2011) study an irrevocable multi-armed bandit model where revisiting to an arm that was pulled but then stopped is disallowed. The model is motivated by application scenarios in which such revisit actions are either unacceptable or too costly. Farias Madan (2011) propose an algorithm for the problem by drawing intuitions from the packing heuristic, and show that the algorithm has a constant performance guarantee.

Following this line of work, Guha Munagala (2013) propose an algorithm for the general finite-horizon Markovian bandit problem. The algorithm satisfies the irrevocability restrictions and obtains a 1/2 performance ratio guarantee. Later, Ma (2018) proposes a generalized model that bridges the finite-horizon Markovian bandit problem with the stochastic knapsack problem in Dean et al. (2008). That paper provides an algorithm for the generalized model, which also obtains a 1/2 ratio under the restrictions of irrevocable policies.

Another line of work on Markovian bandit problems focuses on studying restrictive bandit policies under various types of budget constraints. For instance, Guha Munagala (2009) consider a Markovian bandit model where both the action of pulling an arm and the action of switching between arms incur costs, and these two types of costs are respectively bounded by different budgets. Moreover, that paper considers two types of objectives: maximizing total accumulative reward (past utilization) and identifying the best arm (future utilization). The authors provide algorithms with constant performance ratios in both cases.

Guha Munagala (2013) then summarize a series of budget-constrained Markovian bandit models in Guha Munagala (2007), Guha Munagala (2008) and Guha Munagala (2009). More importantly, that paper proposes a general algorithm framework for solving finite-horizon Markovian bandit problems with various side constraints. In one of the restrictive settings, one has to pull the arms irrevocably and follow the order that is arbitrarily fixed at the beginning. In this setting, that paper provides an algorithm that can always obtain at least 1/4 the expected reward of the optimal policy.

This paper considers a Markovian bandit model where each arm corresponds to a pair of a resource and a time period. Specifically, in each time period, pulling an arm represents matching a resource to a customer that arrives in this period. In addition, the matching decision in each time period is modeled as a set of independent arms. It is worth mentioning that our model with a single time period corresponds to the model in Farias Madan (2011) without irrevocability constraints. Moreover, due to the nature of the matching process, our model with a single resource corresponds to the restrictive setting in Guha Munagala (2013) where the order of pulling different arms is arbitrarily fixed at the beginning. We consider our model a generalization of these two models to multiple resources and multiple time periods. Furthermore, in the single-resource case, we devise an innovative algorithm that improves the 1/4 ratio provided in Guha Munagala (2013). In the generalized case, we integrate this algorithm with the algorithm for the single-period model, and provide a powerful algorithm with a strong performance guarantee.

The Markovian bandit model in this paper also features a non-stationary learning environment where the actions of matching the same resource in different time periods are formulated as independent arms. This setting is less conservative than that of the "restless bandits" where an arm's state keeps evolving regardless of the arm's being pulled or not. We refer the readers to Whittle (1980), Bertsimas Niño-Mora (2000), Nino-Mora (2001) and Guha et al. (2010) for a detailed treatment of the restless bandit problem. In addition, our non-stationary setting is different from the setting in Besbes et al. (2014), Besbes et al. (2015) and Cheung et al. (2019). In those papers, the authors consider a non-stationary reward structure for stochastic bandits, in which the variation of the arms' mean rewards over the entire time horizon is bounded by a variation budget. In this paper, we assume no such variation budgets. In fact, in our model, the rewards of each resource's associated arms in different time periods can vary dramatically. This corresponds to the observation in practice that the popularity of a live web-streaming channel in the evening is significantly higher than that in the morning. We show in this paper that our algorithm's performance guarantee applies to all levels of reward variations.

2.2 Model Formulation

Motivated by the online traffic allocation problem, we consider an online matching model with a Bayesian learning environment. Throughout this paper, we use $[k]$ to denote the set $\{1, 2, \ldots, k\}$ for any positive integer k.

2.2.1 Online Matching Process

We need to match N resources to users arriving in T time periods. Each resource $i \in [N]$, as illustrated in Figure 2-1, has a budget B_i that specifies the maximum number of users (i.e., impressions) it can receive across the horizon. In practice, this number of impressions is given to the central platform from the beginning, and hence we assume each resource i's budget B_i is known a priori.

We consider each time period $t \in [T]$ as a period of time in a day or a week. For example, to model a daily traffic allocation task, we can divide a day into three time periods: morning, afternoon and evening. With such division of time periods, our model can capture the possible variations of each channel's click-through rate during a day. In each time period, a group of D_t users arrive sequentially. Given that the real traffic volume in each time period is high, the central platform can usually estimate the value of D_t accurately, i.e., with small variances. Therefore, we assume that D_t is known for all $t \in [T]$ at the beginning.

The reward of resource i in period t is denoted by $p_{i,t} \in [0, 1]$, which corresponds to the click-through rate of channel i in period t. For each arriving user, we need to match her to one of the resources, or irrevocably reject her (by not displaying any ad or recommendation to her in order to save budgets for future periods [2]). When we

[2]It is not practical to reject users as it wastes valuable impressions. In practice, there are often resources with large budget but small reward values. Then we can always replace the "reject" option by matching users to those resources. Nevertheless, we keep the "reject" option so that the model is clean.

match a user to resource i in period t, we earn reward $p_{i,t}$ in expectation if resource i still has a positive remaining budget. After that, the budget value of the matched resource is depleted by one, regardless of whether it is clicked or not. The goal is to maximize the total reward collected from all the users and resources.

2.2.2 Bayesian Online Learning

Due to the challenges of estimating click-through rates in practice, we assume in our model that the reward values $p_{i,t} \in [0, 1]$ are *unknown* from the beginning. We assume $p_{i,t}$ are drawn from certain probability distributions, and we know these distributions a priori. We then need to learn the true reward values on the fly from real-time observations of users' clicks. Each time we match a user to a resource, we can immediately observe an independent Bernoulli response of the user, which in practice corresponds to the click or no-click action. The success rate of the Bernoulli random variable is $p_{i,t}$ for resource i in period t.

We assume that all the $p_{i,t}$ are independent of each other, because as mentioned in the introduction, the estimation models (e.g., deep learning models) implemented by the traffic allocation platform may refresh the estimation results from time to time using new datasets. Consequently, we can view the combination (i, t) of each resource $i \in [N]$ and time period $t \in [T]$ as an "arm". Pulling arm (i, t) corresponds to matching a user to resource i in period t. After each pull of arm (i, t), we update the posterior probability distribution of $p_{i,t}$ based on the corresponding Bernoulli outcome.

For each arm (i, t), we can use a Markov decision process (MDP) to describe the updating procedure of the posterior distribution of $p_{i,t}$ and the associated decision-making process. Since $p_{i,t}$ is a Bernoulli success rate, we can use two numbers (a, b), which denote respectively the number of successes and failures we have observed, to represent the current state, i.e., knowledge of $p_{i,t}$. Let $S_{i,t}$ denote the state space for (a, b). The initial state of the MDP is $(0, 0)$. For each state $u = (a, b) \in S_{i,t}$, let $\theta_{i,t}^{(u)}(\cdot)$ denote the density function of the posterior distribution of $p_{i,t}$ in state u, and $p_{i,t}^{(u)}$ the expected value of $p_{i,t}$ in state u. Given the posterior probability distribution of $p_{i,t}$, we know

$$p_{i,t}^{(u)} := \int_0^1 p \cdot \theta_{i,t}^{(u)}(p)\, dp.$$

Let $q_{i,t}^{(u,v)}$ denote the transition probability from state u to state v when the arm is pulled in state u. Given $u = (a, b)$, we have

$$q_{i,t}^{(u,v)} = \begin{cases} p_{i,t}^{(u)},\ v = (a+1, b) \\ 1 - p_{i,t}^{(u)},\ v = (a, b+1) \\ 0,\ \text{otherwise.} \end{cases}$$

At the beginning of each period t, the MDPs of arms $(1,t), (2,t), \ldots, (N,t)$ are in state $(0,0)$, as we have not collected any online information about $p_{1,t}, p_{2,t}, \ldots, p_{N,t}$. For no more than D_t times in period t, we need to pull one of the arms $(1,t)$, $(2,t)$, $\ldots$, (N,t) with positive remaining budget. Each time we pull an arm (i,t), we collect reward $p_{i,t}^{(u)}$ in expectation. Then the MDP of arm (i,t) moves from the current state $u \in S_{i,t}$ to a new state $v \in S_{i,t}$ according to transition probability $q_{i,t}^{(u,v)}$. For the arms that are not pulled, their MDPs stay at the same state with probability one and the central platform collects no reward from these arms.

As a summary of the model description, we remark that we can make every matching decision based on the current states of the all the MDPs. The estimation of $p_{i,t}$ becomes more accurate as we allocate more users to resource i in period t. After making a matching decision in period t and observing the corresponding Bernoulli response, we need to make a choice among the following decisions:

1. match the same resource to the next user in period t;

2. match another resource to the next user in period t;

3. reject all remaining users in period t and save the budgets of all resources for future time periods.

The central platform's learning and decision-making process characterizes the following challenges. First, the platform faces the classic trade-off between learning (i.e., exploration) and earning (i.e., exploitation) in each time period. Specifically, the platform needs to balance the number of displays between the matching options with established high rewards (exploitation) and the matching options whose rewards are potentially high but still uncertain (exploration). Second, the platform needs to decide how to allocate each resource's budget among different time periods. The decision on whether to allocate the budget to the current time period or save the budget for future time periods reinforces the importance of the balance between exploitation and exploration.

Performance guarantee. In principle, we can use a dynamic program to find the optimal algorithm for maximizing the total expected reward, on the joint state space of the $N \times T$ arms. Unfortunately, this approach suffers from the curse of dimensionality. Thus, in this paper, we aim to design an approximation algorithm with a provable performance guarantee.

To be more precise, consider a problem instance $\mathcal{I}$, which specifies the number of resources N, the number of time periods T, budget constraint B_i, demand size D_t, and the prior distribution $\theta_{i,t}$ of reward $p_{i,t}$, for each arm (i,t) with $i \in [N]$ and $t \in [T]$. Let $\mathsf{ALG}(\mathcal{I})$ denote the expected reward of our algorithm for problem instance $\mathcal{I}$, and $\mathsf{OPT}(\mathcal{I})$ the expected reward of the optimal algorithm. (The expectations are taken over the randomness in the MDP transitions and an algorithm's decisions.) We propose and analyze online algorithms with the following strong guarantee:

$$\mathsf{ALG}(\mathcal{I}) \geq \alpha \cdot \mathsf{OPT}(\mathcal{I}) \text{ for all } \mathcal{I} \tag{2.1}$$

where α is a constant to be proved.

LP relaxation. To provide an analytical benchmark in place of OPT, we formulate a linear program relaxation of the Markovian problem.

In this formulation, $y_{i,t}^{(u)} \in [0,1]$ represents the probability that the MDP of arm (i,t) ever "enters" state u; $x_{i,t}^{(u)} \in [0,1]$ represents the probability that the MDP of arm (i,t) ever enters state u and an algorithm pulls arm (i,t) while its corresponding MDP is in state u. More specifically, for arm (i,t), $y_{i,t}^{(u)}$ is defined on states $u \in S''_{i,t}$ where $S''_{i,t} = \{(a,b) \mid a+b \leq \min\{B_i, D_t\}\}$, and $x_{i,t}^{(u)}$ is defined on states $u \in S'_{i,t}$ where $S'_{i,t} = \{(a,b) \mid a+b < \min\{B_i, D_t\}\}$. Constraints (2.4) and (2.5) describe the budget constraints and group-size constraints. Constraint (2.8) ensures that each MDP starts from the initial state $\rho := (0,0)$; constraint (2.9) ensures that the transition probabilities in each MDP are valid following our definition of $x_{i,t}^{(u)}$ and $y_{i,t}^{(u)}$.

$$\mathsf{LP} := \max \sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{u\in S'_{i,t}} p_{i,t}^{(u)} x_{i,t}^{(u)}. \tag{2.2}$$

$$\text{s.t. } 0 \le x_{i,t}^{(u)} \le y_{i,t}^{(u)}, \ \forall\, u \in S'_{i,t} \ \ t\in[T] \ \ i\in[N] \tag{2.3}$$

$$\sum_{t=1}^{T}\sum_{u\in S'_{i,t}} x_{i,t}^{(u)} \le B_i, \ \forall\, i\in[N] \tag{2.4}$$

$$\sum_{i=1}^{N}\sum_{u\in S'_{i,t}} x_{i,t}^{(u)} \le D_t, \ \forall\, t\in[T] \tag{2.5}$$

$$x_{i,t}^{(u)} \in [0,1], \ \forall\, u\in S'_{i,t} \ \ t\in[T] \tag{2.6}$$

$$y_{i,t}^{(u)} \in [0,1], \ \forall\, u\in S''_{i,t} \ \ t\in[T] \tag{2.7}$$

$$y_{i,t}^{(\rho)} = 1, \ \forall\, t\in[T] \ \ i\in[N] \tag{2.8}$$

$$y_{i,t}^{(v)} = \sum_{u\in S'_{i,t}} x_{i,t}^{(u)} q_{i,t}^{(u,v)}, \ \forall\, v\in S''_{i,t}\setminus\{\rho\} \ \ t\in[T] \ \ i\in[N] \ . \tag{2.9}$$

This linear program provides an upper bound on the expected reward of the optimal algorithm. We state this result in Theorem 2.1. We use LP to denote both the linear program and its optimal value. A similar linear program formulation is also proposed in paper Farias Madan (2011) and Guha Munagala (2013).

Theorem 2.1. *The expected reward of the optimal algorithm is upper-bounded by LP.*

2.2.3 Preliminaries

We introduce in this section the basic technical details for designing our algorithm and proving the algorithm's performance guarantee. These technical details are based on the results developed in Guha Munagala (2013).

Notice that the solution to LP does not directly correspond to an algorithm for our model since the decision variables do not capture the joint evolution of the states of different arms. Nevertheless, the solution can be used to construct a collection of "de-coupled policies" for each single arm, and the ideas of Guha Munagala (2013) for devising approximation algorithms are based on “assembling” these de-coupled policies into online algorithms. Our novel algorithms in Section 2.3 and Section 2.4 are also built upon assembling such de-coupled policies.

We call the decoupled policies *single-arm policies*. Specifically, for each arm (i, t), we denote the arm's *single-arm policy* as $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)$ and it has the following parameters:

- feasible solution $\mathbf{x}_{i,t} = \{x_{i,t}^{(u)}, u \in S'_{i,t}\}$, $\mathbf{y}_{i,t} = \{y_{i,t}^{(u)}, u \in S''_{i,t}\}$ to LP;
- allowed number of pulls or “budget" parameter K.

The single-arm policy $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)$ for each arm (i, t) specifies a (randomized) mapping from each state $u = (a, b)$ in $S_{i,t}$ with $a + b < K$ to one of the decisions: (i) pull the arm, or (ii) stop. Formally, we define $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)$ as follows.

```
Single-arm policy $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)$
Initialization: set $a = b = 0$ and state $u := (a, b)$
While $a + b < K$:
    Choose $w \in [0, y_{i,t}^{(u)}]$ uniformly at random:
        (a). If $w \in [0, x_{i,t}^{(u)}]$, then pull the arm $(i, t)$.
             Observe a transition of the MDP from $u$ to $v := (a', b')$.
             Set $u \leftarrow v$, $a \leftarrow a'$, $b \leftarrow b'$.
        (b). If $w \in (x_{i,t}^{(u)}, y_{i,t}^{(u)}]$, then stop.
Return $a + b$.
```

We can interpret single-arm policies as stopping trials. For each arm (i, t), we keep pulling the arm until the number of pulls reaches budget K, or until we observe $w \in (x_{i,t}^{(u)}, y_{i,t}^{(u)}]$ in state u. The policy then returns the total number of pulls as a result.

Let $\mathcal{R}(\cdot)$ and $\mathcal{K}(\cdot)$ be the *expected reward* and *expected cost* (i.e., number of pulls) functions of a single-arm policy, respectively. Given single-arm policy $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)$, we have

$$\mathcal{R}(\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)) := \sum_{u:(a,b)\in S'_{i,t}} p_{i,t}^{(u)} x_{i,t}^{(u)} \mathbf{1}(a+b<K), \tag{2.10}$$

$$\mathcal{K}(\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, K)) := \sum_{u:(a,b)\in S'_{i,t}} x_{i,t}^{(u)} \mathbf{1}(a+b<K). \tag{2.11}$$

The summations in (2.10) and (2.11) are taken over the states (a, b) in $S'_{i,t}$ where $a + b < \min\{B_i, D_t\}$.

Now we consider the "assembling procedure" that runs a collection of single-arm policies according to a pre-specified order and an adaptively updated budget parameter. This assembling procedure defines a general algorithm framework for designing algorithms with single-arm policies, and we will frequently use this framework throughout this paper. To illustrate the idea of this general framework, we consider M arms, with each arm having a budget parameter U_m. In addition, these M arms share a common budget W.

The assembling procedure takes the following components as inputs:

- single-arm policies: $\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, \cdot)$ for $m \in [M]$;
- an order of pulling the M arms (i.e., running the corresponding single-arm policies): $I(1), \ldots, I(M)$, which is a permutation of $1, \ldots, M$.

The framework is formally defined as follows. Note that the algorithm framework is a central component for developing the algorithms for our problem where each *coupling* constraint (with respect to B_i or D_t) corresponds to an assembling procedure. Specifically in our problem, for each period $t \in [T]$, we have N arms $(1, t), \ldots, (N, t)$ sharing budget D_t, and for each resource $i \in [N]$, we have T arms $(i, 1), \ldots, (i, T)$ sharing budget B_i. Therefore, given single-arm policies $\mathcal{P}_{i,t}(\mathbf{x}_{i,t}, \mathbf{y}_{i,t}, \cdot)$, we can devise our algorithm by applying Framework to the single-arm policies of these different sets of arms.

In the algorithm framework, we keep updating the budget parameter of each single-arm policy. Given single-arm policy $\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, K)$, when its budget parameter is reduced to βK with $\beta \in [0, 1]$, we provide in Theorem 2.2 the relationship between

$\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, K)$ and $\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, \beta K)$ in terms of their expected rewards and expected costs. Based on this result, we show in Proposition 2.1 a lower bound of the expected reward of any algorithm that fits into Framework.

```
Framework
Input: single-arm policies $\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, \cdot)$, $m \in [M]$;
        order $I(1), ...., I(M)$.
Initialization: set $W' = W$.
For $k = [1, \ldots, M]$:
        $n \leftarrow \mathcal{P}_{I(k)}(\mathbf{x}_{I(k)}, \mathbf{y}_{I(k)}, \min\{U_{I(k)}, W'\})$;
        $W' \leftarrow W' - n$.
```

Theorem 2.2. *Given single-arm policy $\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, K)$ with budget parameter (i.e., allowed number of pulls) K, if K is reduced to βK where $\beta \in [0, 1]$, then we have*

$$\begin{aligned} (i)\ \mathcal{R}\left(\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, \beta K)\right) &\geq \beta \cdot \mathcal{R}\left(\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, K)\right), \\ (ii)\ \mathcal{K}\left(\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, \beta K)\right) &\leq \mathcal{K}\left(\mathcal{P}_m(\mathbf{x}_m, \mathbf{y}_m, K)\right). \end{aligned}$$

Proposition 2.1. *Consider M arms that share budget W, and each arm $m \in [M]$ has budget U_m. Given single-arm policies $\mathcal{P}_m(\cdot, K_m)$ with budget parameter $K_m = \min\{U_m, W\}$ for each $m \in [M]$, expected reward $\mathcal{R}'_m = \mathcal{R}(\mathcal{P}_m(\cdot, K_m))$ and expected cost $\mathcal{K}'_m = \mathcal{K}(\mathcal{P}_m(\cdot, K_m))$ then the expected reward of any algorithm that runs these single-arm policies with Framework in the order $I(1), \ldots, I(M)$ is at least*

$$\sum_{k \in [M]} \left(1 - \frac{1}{W} \sum_{j<k} \mathcal{K}'_{I(j)}\right) \mathcal{R}'_{I(k)}. \tag{2.12}$$

2.3 Algorithms for Two Special Cases

We consider in this section two special cases of the online resource allocation problem, where we assume $T = 1$ (i.e., single time period) and $N = 1$ (i.e., single resource), respectively. The algorithms for these two special cases are both based on Framework.

More importantly, the design of these two algorithms motivate the idea for devising the algorithm for the general problem where T and N can be any positive integers.

The single-period problem where $T = 1$ has been well studied in Farias Madan (2011) and Guha Munagala (2013). For convenience of discussion, we restate the algorithm using Framework and provide the main results from Guha Munagala (2013) in Section 2.3.1.

The single-resource problem where $N = 1$ is the major focus of our paper. In this problem, we cannot decide the order of pulling different arms, but have to follow the time order, and therefore, the algorithm for the $T = 1$ model cannot be applied. To tackle this challenge, we propose a novel algorithm that judiciously allocates the budget between the "good" arms that have relatively high rewards and the "bad" arms that have relatively low rewards. We present the development of the algorithm and the algorithm's performance guarantee in Section 2.3.2.

2.3.1 Single time period and multiple resources

In the case of $T = 1$, we have N arms sharing budget D (subscript t omitted), and each arm i has budget B_i for $i \in [N]$. Given single-arm policy $\mathcal{P}_i(\cdot) := \mathcal{P}_i(\mathbf{x}_i^*, \mathbf{y}_i^*, \cdot)$ with optimal solution $\mathbf{x}_i^*$, $\mathbf{y}_i^*$ to LP, let $\mathcal{R}_i$ and $\mathcal{K}_i$ be the expected reward and expected cost of the policy $\mathcal{P}_i(\min\{B_i, D\})$ with budget parameter $\min\{B_i, D\}$.

To solve this problem, we follow the "greedy" approach proposed in Farias Madan (2011) and Guha Munagala (2013) that runs single-arm policies $\mathcal{P}_i(\cdot)$ for $i \in [N]$ with Framework in a decreasing order of their "reward-to-cost ratios". We refer to the algorithm as Greedy, and describe its detailed procedure as follows.

We obtain from Guha Munagala (2013) that

$$\sum_{k\in[N]} \left(1 - \frac{1}{D}\sum_{j<k}\mathcal{K}_{I(j)}\right)\mathcal{R}_{I(k)} \geq \frac{1}{2}\sum_{i\in[N]}\mathcal{R}_i. \tag{2.13}$$

By Proposition 2.1, we know the left-hand side of (2.13) is a lower bound of the expected reward of algorithm Greedy. In addition, by Theorem 2.1, we know that

$\sum_{i\in[N]} \mathcal{R}_i = \mathsf{LP} \geq \mathsf{OPT}$, where OPT denotes the expected reward of the optimal algorithm for the problem. Let ALG denote the expected reward of algorithm Greedy, and we obtain the performance guarantee $\mathsf{ALG} \geq \frac{1}{2}\mathsf{OPT}$.

```
Greedy
Input: single-arm policies $\mathcal{P}_i(\cdot)$ for $i \in [N]$;
        order $I(1), ...., I(N)$ such that

            $$\frac{\mathcal{R}_{I(1)}}{\mathcal{K}_{I(1)}} \geq \frac{\mathcal{R}_{I(2)}}{\mathcal{K}_{I(2)}} \geq \cdots \geq \frac{\mathcal{R}_{I(N)}}{\mathcal{K}_{I(N)}}.$$

Initialization: set $D' = D$.
For $k$ in $[1, \ldots, N]$:
        $n \leftarrow \mathcal{P}_{I(k)}(\min\{B_{I(k)}, D'\})$;
        $D' \leftarrow D' - n$.
```

2.3.2 Single resource and multiple time periods

In the case of $N = 1$, we have T arms sharing budget B (subscript i omitted), and each arm t has budget D_t. This problem is much more challenging than the previous one due to an additional restriction that we must pull the arms according to the time order $1, \ldots, T$. Hence the previous greedy algorithm, in which we have the freedom of choosing the order of pulling different arms, cannot be applied to this case.

To tackle this challenge raised by the time order restriction, we develop in this paper an innovative algorithm that differentiates the allocation of each arm's budget based on the arm's relative reward levels. The outline for developing the algorithm is as follows. First, we describe the worst-case input order for any algorithm that uses Framework. Then with this worst-case order, we separate the set of high-rewarding arms and the set of low-rewarding arms by judiciously selecting a threshold value of the single-arm policy's reward-to-cost ratio. Last, we modify each arm's single-arm policy with differentiation based on the set each arm belongs to. We propose the algorithm that runs these modified policies with Framework in the fixed time order, and we prove our algorithm's performance guarantee by establishing a condition that is similar to (2.13).

Worst-case Order

The worst-case order for any algorithm that runs under Framework is defined in terms of the lower bound of the algorithm's expected reward, and its formal definition is provided in Proposition 2.2.

Proposition 2.2. *The lower bound* (2.12) *shown in Proposition 2.1 is minimized when the order* $I(1), \ldots, I(M)$ *is such that*

$$\frac{\mathcal{R}'_{I(1)}}{\mathcal{K}'_{I(1)}} \leq \frac{\mathcal{R}'_{I(2)}}{\mathcal{K}'_{I(2)}} \leq \ldots \leq \frac{\mathcal{R}'_{I(M)}}{\mathcal{K}'_{I(M)}}. \tag{2.14}$$

We refer to the order $I(1), \ldots, I(M)$ *that satisfies* (2.14) *as the* ***worst-case order****.*

In particular, given single-arm policy $\mathcal{P}_t(\cdot) := \mathcal{P}_t(\mathbf{x}_t^*, \mathbf{y}_t^*, \cdot)$, expected reward $\mathcal{R}_t$ and expected cost $\mathcal{K}_t$ of the policy $\mathcal{P}_t(\min\{B, D_t\})$, consider the algorithm that runs single-arm policies $\mathcal{P}_t(\cdot)$ with Framework in order $t = 1, \ldots, T$. By Proposition 2.2, in the worst case, the reward-to-cost ratios of these single-arm policies increase in t, namely, $\mathcal{R}_1/\mathcal{K}_1 \leq \ldots \leq \mathcal{R}_T/\mathcal{K}_T$, and the value of the lower bound (2.12) is now

$$\sum_{t \in [T]} \left(1 - \frac{1}{B} \sum_{s<t} \mathcal{K}_s\right) \mathcal{R}_t. \tag{2.15}$$

We illustrate the lower bound for the worst-case order in Figure 2-2. We observe that the value of (2.15) is equivalent to the area of the highlighted rectangles in the figure. We can also lower-bound this value by the area under the piecewise-linear curve where the slope of the t-th line segment is given by $B \cdot (\mathcal{R}_t/\mathcal{K}_t)$. Specifically in Figure 2-2, from bottom to top, the area of the first rectangle is $\mathcal{R}_1$, and the second $(1 - \mathcal{K}_1/B)\,\mathcal{R}_2$, etc. In addition, the total accumulative height of the rectangles is equal to $\sum_{t \in [T]} \mathcal{R}_t$, so is the total area of the underlying region.

With abuse of notations, let OPT denote the expected reward of the optimal algorithm for this single-resource problem, and ALG the expected reward of the algorithm that runs $\mathcal{P}_t(\cdot)$ with $\mathsf{Framework}$. By Theorem 2.1, we know $\sum_{t\in[T]}\mathcal{R}_t = \mathsf{LP} \geq \mathsf{OPT}$. The algorithm's performance guarantee in terms of $\mathsf{ALG}/\mathsf{OPT}$ is hence lower-bounded by the ratio between the total area of the rectangles (*or* the area under the piecewise-linear curve) and the area of the underlying region.

Suppose in the worst-case order, the reward-to-cost ratio $\mathcal{R}_T/\mathcal{K}_T$, is much larger than the ratios of the other single-arm policies. Then in Figure 2-2, the slope of piecewise-linear curve is small for the first $T-1$ segments, and then increases significantly for the T-th segment. Correspondingly, the area under the curve would be very small, and it is thus difficult to find a constant that lower-bounds the algorithm's performance guarantee $\mathsf{ALG}/\mathsf{OPT}$. In other words, the algorithm would use most of the budget pulling the less rewarding arms, and such inefficient use of the budget makes it difficult to establish a condition similar to (2.13).

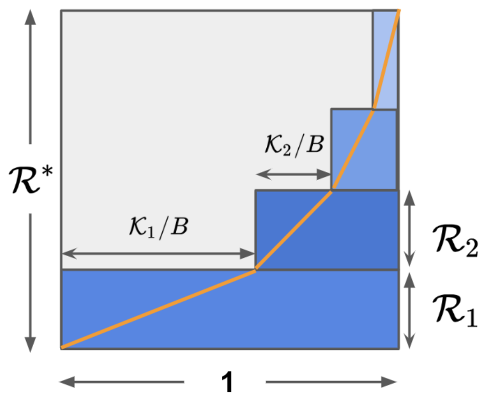


Figure 2-2: A pictorial explanation of the lower bound of an algorithm's expected reward under $\mathsf{Framework}$.

Cutting Point

We differentiate the arms by selecting a threshold value of the reward-to-cost ratios of their single-arm policies.

We re-arrange the arms according to the worst-case order shown in (2.14). For simplicity, we assume without loss of generality that $\mathcal{R}_1/\mathcal{K}_1 \leq \ldots \leq \mathcal{R}_T/\mathcal{K}_T$. Let $R^* := \sum_{t\in[T]}\mathcal{R}_t$. We construct function $F(x)$, which describes the piecewise-linear curve shown in Figure 2-2, with the y-axis scaled by $1/R^*$:

$$F(x) = \frac{1}{R^*}\sum_{s<t}\mathcal{R}_s + \frac{\mathcal{R}_t/R^*}{\mathcal{K}_t/B}\left(x - \frac{1}{B}\sum_{s<t}\mathcal{K}_s\right) \quad \text{for } x \in \left[\frac{1}{B}\sum_{s<t}\mathcal{K}_s, \frac{1}{B}\sum_{s\le t}\mathcal{K}_s\right]. \tag{2.16}$$

We decide the threshold value of the reward-to-cost ratios by finding a "cutting" point on the function curve of $F(x)$. In particular, we observe that the slope of each line segment of $F(x)$, namely, $(\mathcal{R}_t/R^*)\,/\,(\mathcal{K}_t/B)$, is proportional to each single-arm policy's reward-to-cost ratio. Let $F'(x)$ denote the set of sub-gradients of function $F(\cdot)$ at value x. The cutting point (x^*, y^*) with $y^* := F(x^*)$ is subject to the condition

$$1 - F(x^*) \in F'(x^*). \tag{2.17}$$

The condition (2.17) characterizes a judicious rule for selecting the cutting point. Consider an algorithm that saves the budget for the high-rewarding arms by reducing the budget of the low-rewarding arms. Given cutting point (x^*, y^*), if $1 - F(x^*) < F'(x^*)$, then due to the fact that $F(x)$ and $F'(x)$ both increase in x, most arms would be divided into the set of high-rewarding arms. In this case, the algorithm's budget allocation scheme would fail to distinguish the arms by their reward levels. On the other hand, if $1 - F(x^*) > F'(x^*)$, then most of arms would be divided into the set of low-rewarding arms. In this case, the allocation scheme would reserve the budget to only a few arms, and such a conservative allocation would also result in an inefficient use of the total budget. Therefore, the condition $1 - F(x^*) \in F'(x^*)$ finds a perfect balance between the previous two scenarios.

We can use line search methods to find the cutting point that satisfies (2.17). We show in Lemma 2.1 that the cutting point is guaranteed to exist and is unique. The claim is due to the facts: i) $F(x)$ and $F'(x)$ are both continuous in x; ii) $1 - F(x)$ is strictly decreasing in x; and iii) $F'(x)$ is non-decreasing in x.

Lemma 2.1. *Given function $F(x)$ as defined in* (2.16)*, the cutting point (x^*, y^*) that satisfies* (2.17) *is guaranteed to exist and is unique.*

As shown in Figure 2-3, the cutting point defines three different sets of arms.

We denote them as G_0, G_1 and G_2, respectively. Set G_0 contains the arms whose corresponding line segments are below the cutting point, namely,

$$G_0 := \left\{ t : \frac{\mathcal{R}_t / R^*}{\mathcal{K}_t / B} < \underline{F}'(x^*) \right\},$$

where $\underline{F}'(x^*)$ denotes the minimum sub-gradient in set $F'(x^*)$. Set G_1 contains the arm whose corresponding line segment is exactly on the cutting point, namely,

$$G_1 := \left\{ t : \frac{\mathcal{R}_t / R^*}{\mathcal{K}_t / B} = 1 - F(x^*) \right\}.$$

If the cutting point happens to be a breakpoint of the piecewise linear function $F(x)$, then G_1 is empty. Set G_2 contains the arms whose corresponding line segments are above the cutting point, namely,

$$G_2 := \left\{ t : \frac{\mathcal{R}_t / R^*}{\mathcal{K}_t / B} > \overline{F}'(x^*) \right\},$$

where $\overline{F}'(x^*)$ denotes the maximum sub-gradient in set $F'(x^*)$.

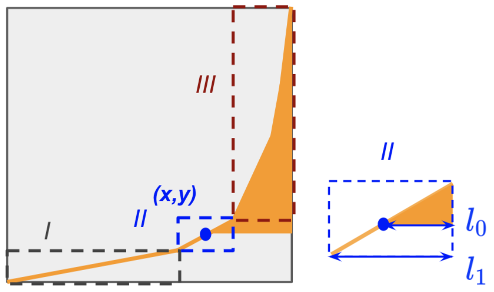


Figure 2-3: Sets of arms divided by the cutting point: G_0(I), G_1(II), G_2(III).

Prophet Policy

We modify each arm's single-arm policy with differentiations based on the set each arm belongs to. For the arms in set G_0, since their single-arm policies have low reward-to-cost ratios, they are "discarded" from the beginning, meaning their single-arm policies are stopped from the root state of their MDPs. For the arms in set G_2, since their single-arm policies have high reward-to-cost ratios, they are pulled according to $\mathcal{P}_t(\cdot)$ with no modifications. For the arm in set G_1 (if it is not empty), since its single-arm policy's reward-to-cost ratio is in-between, the arm is either discarded from the beginning or pulled with a certain probability. Let μ denote the probability that the arm in G_1 is pulled. We define $\mu := l_0/l_1$, where

$$l_0 = \frac{1}{B}\sum_{t\in G_0}\mathcal{K}_t + \frac{1}{B}\sum_{t\in G_1}\mathcal{K}_t - x^*, \tag{2.18}$$

$$l_1 = \frac{1}{B}\sum_{t\in G_1}\mathcal{K}_t. \tag{2.19}$$

Essentially, given the piecewise linear function curve of $F(x)$ and cutting point x^*, the value of μ measures the portion of the arm's corresponding line segment that is above the cutting point. See Figure 2-3 for a detailed illustration.

We call these modified single-arm policies *prophet policies*, and denote them as $\mathcal{P}_t(\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{y}}_t, \cdot)$ following the definition of single-arm policies in Section 2.2.3. The feasible solution $\widetilde{\mathbf{x}}_t$ and $\widetilde{\mathbf{y}}_t$ in $\widetilde{\mathcal{P}}_t(\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{y}}_t, \cdot)$ is modified based on $\mathbf{x}_t^*$ and $\mathbf{y}_t^*$, i.e., the optimal solution to LP. The detailed procedure is described as follows.

Prophet Policy $\widetilde{\mathcal{P}}_t(\cdot)$:

Fix $\mathbf{x}_t = \mathbf{x}_t^*$ and $\mathbf{y}_t = \mathbf{y}_t^*$.

(a). If $t \in G_0$: define $\widetilde{\mathbf{x}}_t = \mathbf{0}$;

define $\widetilde{y}_t^{(u)} = 0$ for all state u in $S_t' \setminus \{\rho\}$, and $\widetilde{y}_t^{(\rho)} = 1$.

(b). If $t \in G_1$: define $\widetilde{\mathbf{x}}_t = \mu \cdot \mathbf{x}_t$ where $\mu = l_0/l_1$ is calculated by (2.18) and (2.19);

define $\widetilde{y}_t^{(u)} = \mu \cdot y_t^{(u)}$ for all state u in $S_t'' \setminus \{\rho\}$, and $\widetilde{y}_t^{(\rho)} = 1$.

(c). If $t \in G_2$: define $\widetilde{\mathbf{x}}_t = \mathbf{x}_t$, and $\widetilde{\mathbf{y}}_t = \mathbf{y}_t$.

Return $\mathcal{P}_t(\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{y}}_t, \cdot)$

Given prophet policy $\widetilde{\mathcal{P}}_t(\cdot) := \widetilde{\mathcal{P}}_t(\widetilde{\mathbf{x}}_t, \widetilde{\mathbf{y}}_t, \cdot)$, let $\widetilde{\mathcal{R}}_t$ and $\widetilde{\mathcal{K}}_t$ denote the expected reward and expected cost of the policy $\widetilde{\mathcal{P}}_t(\min\{B, D_t\})$ with budget parameter $\min\{B, D_t\}$. By definition (2.10) and (2.11), we obtain i) $\widetilde{\mathcal{K}}_t = 0$ and $\widetilde{\mathcal{R}}_t = 0$ for arm t in set G_0,

ii) $\widetilde{\mathcal{K}}_t = \mu \cdot \mathcal{K}_t$ and $\widetilde{\mathcal{R}}_t = \mu \cdot \mathcal{R}_t$ for arm t in set G_1, iii) $\widetilde{\mathcal{K}}_t = \mathcal{K}_t$ and $\widetilde{\mathcal{R}}_t = \mathcal{R}_t$ for arm t in set G_2. With prophet policies, we observe that the cost of the low-rewarding arms in G_0 is reduced to zero, and the total budget is thus reserved for the high-rewarding arms in G_1 and G_2.

We propose algorithm Prophet that runs prophet policies $\widetilde{\mathcal{P}}_t(\cdot)$ with Framework. The detailed procedure of the algorithm is described as follows.

```
Prophet
Input: prophet policies $\widetilde{\mathcal{P}}_t(\cdot)$ for $t \in [T]$;
         order $I(1), ...., I(T)$ with $I(t) = t$
Initialization: set $B' = B$.
For period $t$ in $[1, \dots, T]$:
         $n \leftarrow \widetilde{\mathcal{P}}_{I(t)}(\min\{B', D_{I(t)}\})$;
         $B' \leftarrow B' - n$.
```

Performance Analysis

By Proposition 2.2, we know it suffices to analyze the worst-case order to show the performance guarantee of any algorithm that runs a collection of single-arm policies with Framework. Therefore, in the following analysis, we assume that the time order, i.e., the fixed order of pulling the T arms $I(t) = t$, follows the definition of the worst-case order shown in (2.14).

Let ALG be the expected reward of algorithm Prophet. We analyze the performance guarantee of algorithm Prophet by establishing the equivalence between the area under the *truncated* function curve of $F(x)$ and a lower bound of the ratio ALG/OPT. More specifically, by Proposition 2.1, we know that the performance guarantee ALG/OPT for algorithm Prophet is lower-bounded by

$$\sum_{t \in [T]} \left(1 - \frac{1}{B} \sum_{s<t} \widetilde{\mathcal{K}}_s\right) \widetilde{\mathcal{R}}_t / R^*. \tag{2.20}$$

Following the discussion of Figure 2-2, in the worst-case order, this value is lower-bounded by the area under the truncated function curve of $F(x)$, as shown in Figure 2-4, and furthermore, lower-bounded by the total area of a triangle and a rectangle.

Let e_1 denote the area of the triangle and e_2 the area of the rectangle. Given cutting point (x^*, y^*), where $y^* = F(x^*)$, we can calculate e_1 and e_2 as follows.

$$\text{Triangle: } e_1 = \frac{1}{2}(1-x^*)^2 \cdot \overline{F}'(x^*). \tag{2.21}$$

$$\text{Rectangle: } e_2 = x^* \cdot (1 - F(x^*)). \tag{2.22}$$

We show in Lemma 2.2 that the sum of e_1 and e_2 is lower-bounded by a constant. Based on this result, we provide in Theorem 2.3 the performance guarantee of algorithm Prophet.

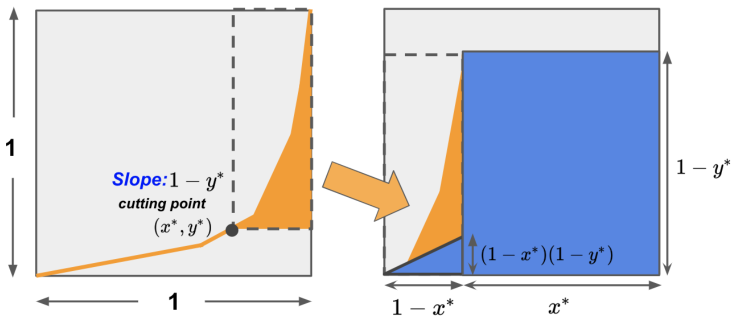


Figure 2-4: A pictorial explanation of the performance guarantee of algorithm Prophet.

Lemma 2.2. *Given function $F(x)$ as defined in (2.16), cutting point $(x^*, F(x^*))$ as defined in (2.17), and e_1 and e_2 as defined in (2.21) and (2.22), we have $e_1 + e_2 \geq \sqrt{2} - 1$.*

Theorem 2.3. *Given prophet policies $\widetilde{\mathcal{P}}_t(\cdot)$, expected reward $\widetilde{\mathcal{R}}_t$ and expected cost $\widetilde{\mathcal{K}}_t$ of the prophet policy $\widetilde{\mathcal{P}}_t(\min\{B, D_t\})$ with budget parameter $\min\{B, D_t\}$ for $t \in [T]$, the expected reward of algorithm Prophet is lower bounded by*

$$\sum_{t\in[T]} \left(1 - \frac{1}{B}\sum_{s<t} \widetilde{\mathcal{K}}_s\right) \widetilde{\mathcal{R}}_t,$$

which is at least $\sqrt{2}-1$ the expected reward of the optimal algorithm. That is,

$$\sum_{t\in[T]}\left(1-\frac{1}{B}\sum_{s<t}\widetilde{\mathcal{K}}_s\right)\widetilde{\mathcal{R}}_t \geq \left(\sqrt{2}-1\right)\sum_{t\in[T]}\mathcal{R}_t.$$

Therefore, for algorithm Prophet, *we have*

$$ALG \geq \left(\sqrt{2}-1\right) OPT.$$

2.4 Algorithm for the General Problem

We present in this section our algorithm for the general problem that is stated in Section 4.2.1, where the number of resources N and the number of time periods T can be any positive integers. In fact, we can decompose the problem into two dimensions. In the time dimension, for each time period $t \in [T]$, given demand size D_t, we have N arms $(\cdot, t)$ sharing budget D_t. In the resource dimension, for each resource $i \in [N]$, given resource budget B_t, we have T arms $(i, \cdot)$ sharing budget B_t. Both dimensions of the problem, as special cases, have been discussed in Section 2.3, and our development of the algorithm for the general problem is based on integrating the algorithms for these two special cases with Framework.

The general problem combines the challenges in both dimensions. Due to budget limit B_i and demand size D_t, the learning and earning trade-off appears both for each resource $i \in [N]$ and in each time period $t \in [T]$. More importantly, for each $i \in [N]$, the order of pulling arms (i, t) has to follow the time order $t = 1, \ldots, T$. Based on the discussion in Section 2.3.2, we know that this restrictive order raises extra difficulties for designing an algorithm with a strong performance guarantee.

To tackle the challenges from both problem dimensions, we divide our algorithm into two stages and conquer each dimension of the problem separately.

In the first stage, we implement a budget allocation scheme for each resource by preparing the prophet policies, as discussed in Section 2.3.2. Given single-arm policy $\mathcal{P}_{i,t}(\cdot) := \mathcal{P}_{i,t}(\mathbf{x}^*_{i,t}, \mathbf{y}^*_{i,t}, \cdot)$, as defined in Section 2.2.3, we obtain expected reward

$\mathcal{R}_{i,t} := \mathcal{R}(\mathcal{P}_{i,t}(\min\{B_i, D_t\}))$ and expected cost $\mathcal{K}_{i,t} := \mathcal{K}(\mathcal{P}_{i,t}(\min\{B_i, D_t\}))$ of the policy $\mathcal{P}_{i,t}(\min\{B_i, D_t\}))$ with budget parameter $\min\{B_i, D_t\})$, as defined in (2.10) and (2.11).

We re-arrange the set of arms $(i, \cdot)$ according to the worst-case order for each $i \in [N]$. For simplicity, we assume that $\mathcal{R}_{i,1}/\mathcal{K}_{i,1} \leq \ldots \leq \mathcal{R}_{i,T}/\mathcal{K}_{i,T}$. Let $R_i^* := \sum_{t\in[T]} \mathcal{R}_{i,t}$. We construct function $F_i(x)$ for each $i \in [N]$ as shown in (2.16), namely,

$$F_i(x) := \frac{1}{R_i^*}\sum_{s<t}\mathcal{R}_{i,s} + \frac{\mathcal{R}_{i,t}/R_i^*}{\mathcal{K}_{i,t}/B_i}\left(x - \frac{1}{B_i}\sum_{s<t}\mathcal{K}_{i,s}\right) \quad \text{for } x \in \left[\frac{1}{B_i}\sum_{s<t}\mathcal{K}_{i,s}, \frac{1}{B_i}\sum_{s\leq t}\mathcal{K}_{i,s}\right]. \tag{2.23}$$

Let $F_i'(x)$ denote the set of sub-gradients of function $F_i(\cdot)$ at value x. Following condition (2.17), we find cutting point (x_i^*, y_i^*) with $y_i^* := F_i(x_i^*)$ on the function curve of $F_i(x)$, which satisfies

$$1 - F_i(x_i^*) \in F_i'(x_i^*). \tag{2.24}$$

For each $i \in [N]$, the cutting point divides the arms $(i, \cdot)$ into three sets: (i) the set of low-rewarding arms $G_{i,0} = \{(i,t) : (\mathcal{R}_{i,t}/\mathcal{K}_{i,t}) \cdot (B_i/R_i^*) < \underline{F}_i'(x_i^*)\}$, where $\underline{F}_i'(x_i^*)$ denote the minimum sub-gradient in set $F_i'(x_i^*)$; (ii) the set of arms on the cutting point $G_{i,1} = \{(i,t) : (\mathcal{R}_{i,t}/\mathcal{K}_{i,t}) \cdot (B_i/R_i^*) = 1 - F_i(x_i^*)\}$; (iii) the set of high-rewarding arms $G_{i,2} = \left\{(i,t) : (\mathcal{R}_{i,t}/\mathcal{K}_{i,t}) \cdot (B_i/R_i^*) > \overline{F}_i'(x_i^*)\right\}$, where $\overline{F}_i'(x_i^*)$ denote the maximum sub-gradient in set $F_i'(x_i^*)$.

We obtain prophet policies $\widetilde{\mathcal{P}}_{i,t}(\cdot)$ via the following modifications.

```
Prophet Policy $\widetilde{\mathcal{P}}_{i,t}(\cdot)$:
Define $\mu_i := l_{i,0}/l_{i,1}$ where
    $l_{i,0} = \frac{1}{B_i}\sum_{t\in G_{i,0}} \mathcal{K}_{i,t} + \frac{1}{B_i}\sum_{t\in G_{i,1}} \mathcal{K}_{i,t} - x_i^*$,
    $l_{i,1} = \frac{1}{B_i}\sum_{t\in G_{i,1}} \mathcal{K}_{i,t}$.
Fix $\mathbf{x}_{i,t} = \mathbf{x}_{i,t}^*$ and $\mathbf{y}_{i,t} = \mathbf{y}_{i,t}^*$.
    (a). If $t \in G_{i,0}$: define $\widetilde{\mathbf{x}}_{i,t} = \mathbf{0}$;
        define $\widetilde{y}_{i,t}^{(u)} = 0$ for all state $u$ in $S'_{i,t} \setminus \{\rho\}$, and $\widetilde{y}_{i,t}^{(\rho)} = 1$.
    (b). If $t \in G_{i,1}$: define $\widetilde{\mathbf{x}}_{i,t} = \mu_i \cdot \mathbf{x}_{i,t}$;
        define $\widetilde{y}_{i,t}^{(u)} = \mu_i \cdot y_{i,t}^{(u)}$ for all state $u$ in $S''_{i,t} \setminus \{\rho\}$, and $\widetilde{y}_{i,t}^{(\rho)} = 1$.
    (c). If $t \in G_{i,2}$: define $\widetilde{\mathbf{x}}_{i,t} = \mathbf{x}_{i,t}$, and $\widetilde{\mathbf{y}}_{i,t} = \mathbf{y}_{i,t}$.
Return $\mathcal{P}_{i,t}(\widetilde{\mathbf{x}}_{i,t}, \widetilde{\mathbf{y}}_{i,t}, \cdot)$
```

The seconds stage of the algorithm takes place after the matching process starts. Given the division of arms, we know the arms in set $G_{i,0}$ are discarded from the beginning. In addition, let $B_i(t)$ be the remaining budget of resource i at the beginning of time period t. In each period $t \in [T]$, the algorithm only needs to consider pulling the set of *active* arms $(\mathcal{Q}_t, t)$, where $\mathcal{Q}_t$ is given by

$$\mathcal{Q}_t := \{i \mid B_i(t) > 0 \text{ and } t \in G_{i,1} \cup G_{i,2}\}. \tag{2.25}$$

Since we can freely choose the order of pulling the arms $(\cdot, t)$ in each time period $t \in [T]$, we can implement algorithm Greedy, as defined in Section 2.3.1, to run the prophet policies of the arms in $(\mathcal{Q}_t, t)$ with Framework. More importantly, in determining the greedy order of running these policies, we need to consider the updated budget parameter $\min\{B_i(t), D_t\}$. Given arm (i,t), let $\mathcal{R}'_{i,t}$ and $\mathcal{K}'_{i,t}$ be the expected reward and expected cost of the prophet policy $\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\})$ with budget parameter $\min\{B_i(t), D_t\}$. We have

$$\widetilde{\mathcal{R}}'_{i,t} := \mathcal{R}\left(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\})\right), \ \widetilde{\mathcal{K}}'_{i,t} := \mathcal{K}\left(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\})\right). \tag{2.26}$$

We refer to our two-stage algorithm as TS-Prophet, and present its detailed procedure as follows.

```
TS-Prophet
Input: prophet policies $\widetilde{\mathcal{P}}_{i,t}(\cdot)$.
Initialization: set $B_i(1) \leftarrow B_i$ for $i \in [N]$.
For period $t$ in $[1, \ldots, T]$:
1. find the set of active arms $(\mathcal{Q}_t, t)$ as defined in (2.25); set $Q_t \leftarrow |\mathcal{Q}_t|$;
2. calculate expected reward $\widetilde{\mathcal{R}}'_{i,t}$ and expected cost $\widetilde{\mathcal{K}}'_{i,t}$ as defined in (2.26);
3. specify the order $I(1), \ldots, I(Q_t)$ of pulling the $Q_t$ arms such that

   $$\frac{\widetilde{\mathcal{R}}'_{I(1),t}}{\widetilde{\mathcal{K}}'_{I(1),t}} \geq \ldots \geq \frac{\widetilde{\mathcal{R}}'_{I(Q_t),t}}{\widetilde{\mathcal{K}}'_{I(Q_t),t}}. \qquad (2.27)$$

For $k$ in $[1, \ldots, Q_t]$:
    $D' \leftarrow D_t$;
    $n \leftarrow \widetilde{\mathcal{P}}_{I(k),t}\left(\min\{B_{I(k)}(t), D'\}\right)$;
    $B_{I(k)}(t+1) \leftarrow B_{I(k)}(t) - n$;
    $D' \leftarrow D' - n$.
```

Performance analysis. The performance analysis of algorithm TS-Prophet can also be decomposed into two dimensions.

Let $\mathsf{ALG}(t)$ be the expected reward of TS-Prophet in each time period $t \in [T]$. We learn from Section 2.3.1 that by using the greedy approach, the value of $\mathsf{ALG}(t)$ is at least 1/2 of the total expected reward of the prophet policies $\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\})$ for $i \in \mathcal{Q}_t$, namely,

$$\mathsf{ALG}(t) \geq \frac{1}{2} \sum_{i \in \mathcal{Q}_t} \widetilde{\mathcal{R}}'_{i,t}. \qquad (2.28)$$

Additionally, for each resource $i \in [N]$, we learn from Section 2.3.2 that by running the prophet policies $\widetilde{\mathcal{P}}_{i,t}(\cdot)$ with Framework, the algorithm obtains the performance guarantee

$$\sum_{t\in[T]} \widetilde{\mathcal{R}}'_{i,t} \geq (\sqrt{2}-1) \sum_{t\in[T]} \mathcal{R}_{i,t}. \tag{2.29}$$

By integrating the results (2.28) and (2.29), we show in Theorem 2.4 that algorithm TS-Prophet can obtain a strong performance guarantee in terms of the ratio ALG/OPT, which is at least

$$(\sqrt{2}-1)/2 \approx 0.21\,.$$

The detailed proof of the theorem is provided in the Appendix.

Theorem 2.4. *The expected reward of algorithm TS-Prophet is at least* $(\sqrt{2}-1)/2$ *the expected reward of the optimal algorithm. That is, for algorithm TS-Prophet, we have*

$$\frac{\mathsf{ALG}}{\mathsf{OPT}} \geq \frac{1}{2}(\sqrt{2}-1). \tag{2.30}$$

2.5 Numerical Experiments

In this section, we present the results of our numerical experiments to verify the performance guarantee of algorithm TS-Prophet, as shown in Theorem 2.4. In addition, we compare the performance of algorithm TS-Prophet with another two algorithms: the benchmark algorithm Benchmark and the randomized algorithm Random.

2.5.1 Alternative algorithms

Algorithms Benchmark and Random both have a two-stage structure similar to algorithm TS-Prophet. When the matching process starts, all these algorithms run the (modified) single-arm policies with Framework using the greedy approach.

The difference of the algorithms lies in the preparation stage. Specifically, algorithm Benchmark makes no modifications of the single-arm policy $\mathcal{P}_{i,t}(\cdot)$ that is based on the optimal solution to LP. Based on our discussion in Section 2.3.1, in the worst-case order, the algorithm would use most of the budget pulling the less rewarding arms, and thus the algorithm can hardly obtain a strong performance guarantee.

Algorithm Random implements the *randomized policy* that runs single-arm policy $\mathcal{P}_{i,t}(\cdot)$ with probability 1/2 and discards the policy, otherwise. The formal definition of the randomized policy is provided as follows.

```
Randomized Policy $\mathcal{P}'_{i,t}(\cdot)$:
Fix $\mathbf{x}_{i,t} = \mathbf{x}^*_{i,t}$ and $\mathbf{y}_{i,t} = \mathbf{y}^*_{i,t}$.
    Define $\widetilde{\mathbf{x}}_{i,t} = \frac{1}{2} \cdot \mathbf{x}_{i,t}$;
    Define $\widetilde{y}^{(u)}_{i,t} = \frac{1}{2} \cdot y^{(u)}_{i,t}$ for all state $u$ in $S'_{i,t} \setminus \{\rho\}$, and $\widetilde{y}^{(\rho)}_{i,t} = 1$.
Return $\mathcal{P}_{i,t}(\widetilde{\mathbf{x}}_{i,t}, \widetilde{\mathbf{y}}_{i,t}, \cdot)$
```

By using these randomized policies, Random uniformly reduces the expected costs of all the single-arm policies by 1/2, and hence achieves balanced budget allocations. Compared to Benchmark, in the worst-case order, this balanced allocation increases the probability of the high-rewarding arms being pulled, and therefore, Random can obtain a good performance guarantee. Guha Munagala (2013) have shown that in the case of $N = 1$, the expected reward of Random is at least 1/4 the expected reward of the optimal policy. Using similar analysis techniques as in TS-Prophet, we can easily generalize this result to the case where N can be any positive integer, and show that the expected reward of Random is at least 1/8 the expected reward of the optimal policy.

2.5.2 Experiment setup

The setting of our numerical experiments involves the following parameters: the number of resources N, the number of time periods T, budget B_i for resource i, group size D_t for time period t, and the prior probability distribution $\theta_{i,t}$ of reward $p_{i,t}$, for all $i \in [N]$ and $t \in [T]$.

To test the performance of the algorithms in different settings, we first fix the values of N, T, B, and then change the value of D to different scales. In addition, we compare two settings of prior distributions. In the first setting, for each $i \in [N]$, the prior distributions $\theta_{i,t}$ for $t \in [T]$ have similar mean values. In the second setting, for each $i \in [N]$, the mean value of $\theta_{i,t}$ spikes at $t = T$. This setting simulates the scenario for the worst-case order, as described in Proposition 2.2.

More specifically, we set $N = 5$, $T = 6$, $B_i = 20$ for all $i \in [N]$, $D_t = 10, 30, 50$ for all $t \in [T]$, respectively. We use $\mathsf{Beta(a,b)}$ as the prior distributions whose mean values are given by $a/(a + b)$. In the first setting, i.e., Setting 1, for all $i \in [N]$ and $t \in [T]$, we set $a = b = 100 + \mathsf{rand}(1 : 5)$, where $\mathsf{rand}(1 : 5)$ generates a random number from $\{1, 2, \ldots., 5\}$. In the second setting, i.e., Setting 2, for $i \in [N]$ and $t \in [T - 1]$, we set $a = 1 + \mathsf{rand}(1 : 5)$, $b = 100 + \mathsf{rand}(1 : 5)$; for $i \in [N]$ and $t = T$, we set $a = 1000 + \mathsf{rand}(1 : 5)$, $b = 100 + \mathsf{rand}(1 : 5)$.

In each experiment setting, we calculate an algorithm's reward by repeatedly running the algorithm multiple times and then averaging the rewards from all these runs. Specifically, after setting up the parameters, we solve the linear program to obtain the optimal solution and the corresponding single-arm policies. Next, during each run, we first draw values of $p_{i,t}$ from their prior distributions, and then run the algorithms TS-Prophet, Benchmark, Random to obtain each algorithm's corresponding reward. We repeat this process a total of 10,000 times, and calculate the average of each algorithm's rewards in all these runs.

2.5.3 Experiment results and analysis

We report the ratio between an algorithm's average reward and the optimal value of the linear program as the algorithm's performance. The results of different algorithms' performances in different settings are presented in Table 2.1 and Table 2.2.

Table 2.1: Comparison of algorithms under Setting 1

	Demand		
	10*ones(T)	30*ones(T)	50*ones(T)
Benchmark	0.79	0.75	0.77
Random	0.45	0.43	0.44
TS-Prophet	0.78	0.75	0.77

Table 2.2: Comparison of algorithms under Setting 2

	Demand		
	10*ones(T)	30*ones(T)	50*ones(T)
Benchmark	0.84	0.66	0.78
Random	0.45	0.40	0.44
TS-Prophet	0.80	0.77	0.86

First of all, we observe from the numerical results that the performance guarantees of algorithms Random and TS-Prophet are verified: Random can always obtain a ratio of at least $1/8$, and TS-Prophet a ratio of at least $(\sqrt{2}-1)/2 \approx 0.21$. The ratios shown in these numerical results are much higher than the corresponding theoretical lower bounds, since the lower bounds present the performance guarantees in the worst cases.

Second, we observe that Benchmark and TS-Prophet outperform Random in all test settings. To explain this observation, we notice from the design of Random that the algorithm uniformly reduces the expected costs and expected rewards by $1/2$ of all the single-arm policies. Although this uniformly randomized approach provides a balanced budget allocation scheme and a worst-case performance guarantee, it loses a large portion of the reward, and therefore results in the algorithm's poor performance in practice.

Last, we observe that algorithms Benchmark and TS-Prophet have comparable performances in Setting 1, while in Setting 2, TS-Prophet outperforms Benchmark. In fact, in Setting 1, since all the arms have similar reward levels, the budget allocation schemes of both algorithms work similarly. However, in the Setting 2, since the rewards of the arms in the last time period are much higher than those of the rest of the arms, the difference between the two algorithms becomes significant. Specifically, Benchmark does not reserve the budget for the high-rewarding arms in the last period, while TS-Prophet does, and such a difference contributes to the superior performance of TS-Prophet.

The results in Table 2.3 to Table 2.6 further break down an algorithm's performance by presenting the average numbers of pulls from different sets of arms.

Specifically, for Setting 1, Table 2.3 shows the number of pulls in each time period, namely, from arms $(\cdot, t)$, and Table 2.4 shows the number of pulls from each resource, namely, from arms $(i, \cdot)$. Similarly, for Setting 2, Table 2.5 shows the number of pulls in each time period and Table 2.6 shows the number of pulls from each resource. The results in these tables are from the setting where $D_t = 50$ for all $t \in [T]$, and hence the budgets are tight relative to the sizes of demand groups.

In Setting 1, we observe that algorithm Random has the least number of pulls on average. The algorithm only makes a few pulls (i.e., rejects a large percentage of the users) in each time period, as shown in Table 2.3, and only uses a small fraction of each resource's budget, as shown in Table 2.4. Since all the arms have similar reward levels, this inefficient use of the limited number of pulls explains the algorithm's bad performance, as shown in Table 2.1. On the other hand, algorithms Benchmark and TS-Prophet both use around 75% of the total budget $\sum_{i\in[N]} B_i = 100$ on average, so these algorithms perform better than Random.

In Setting 2, we observe that algorithms Random and TS-Prophet both use a small fraction of the total budget on average. The difference, as shown in Table 2.5, is that TS-Prophet makes fewer pulls in the first $T-1$ time periods in order to save the budget for the last period, while algorithm Random uniformly reduces the number of pulls in all time periods. Since the arms in the last period are much more rewarding, TS-Prophet obtains a better performance than Random. In addition, we observe that algorithm Benchmark has the highest number of pulls on average. However, most of these pulls are from the less rewarding arms in the first $T-1$ periods. According to Table 2.5, Benchmark uses fewer pulls in the last period than algorithm TS-Prophet, and this explains the performance gap between Benchmark and TS-Prophet, as shown in Table 2.2.

We can further understand the performance comparison between TS-Prophet and Benchmark by visualizing the process of finding the cutting points in TS-Prophet. Figure 2-5 and Figure 2-6 present the function curves of $F_i(x)$ for all $i \in [N]$ in the two settings of prior distributions.

In Setting 1, as shown in Figure 2-5, since all arms have similar reward levels, the

Table 2.3: Average number of pulls from each period in Setting 1

	Time period t						
	1	2	3	4	5	6	Total
Benchmark	10.086	26.053	5.099	14.882	9.450	9.377	74.947
Random	4.971	13.795	2.768	9.302	6.253	6.774	43.863
TS-Prophet	10.173	26.029	5.087	14.771	9.412	9.350	74.821

Table 2.4: Average number of pulls from each resource in Setting 1

	Resource i					
	1	2	3	4	5	Total
Benchmark	14.789	14.637	15.841	14.935	14.746	74.947
Random	8.608	8.746	9.034	8.820	8.656	43.863
TS-Prophet	14.695	14.581	15.909	14.928	14.708	74.821

line segments in each function curve $F_i(x)$ have similar slopes, which is approximately 1. In such cases, the cutting points, as defined by condition (2.17), are close to the origin, and all arms will be divided into the high-rewarding set. In addition, the prophet policies $\widetilde{\mathcal{P}}_{i,t}(\cdot)$ are very similar to the original single-arm policies $\mathcal{P}_{i,t}(\cdot)$, and therefore TS-Prophet and Benchmark have comparable performances.

In Setting 2, as shown in Figure 2-6, the slopes of the line segments in $F_i(x)$ demonstrate significant differences. This is due to the high rewards of arms $(\cdot, T)$ and the associated high reward-to-cost ratios of single-arm policies $\mathcal{P}_{i,T}(\cdot)$ for $i \in [N]$. Specifically in this case, for resource $i = 2, 3, 5$, the corresponding cutting points separate the arms into sets with significantly different reward levels, i.e., a low-rewarding set and a high-rewarding set. Based on such separation and the associated modifications of the single-arm policies, the judicious budget allocation scheme in algorithm TS-Prophet can reserve the budget for the high-rewarding arms. This explains the differences in the number of pulls, as shown in Table 2.5, and also, the outstanding performance of algorithm TS-Prophet, as shown in Table 2.2.

Table 2.5: Average number of pulls from each time period in Setting 2

	Time period t						
	1	2	3	4	5	6	Total
Benchmark	10.393	20.362	0.000	8.221	5.058	37.106	81.141
Random	5.222	10.619	0.000	4.117	2.936	21.750	44.644
TS-Prophet	0.016	0.013	0.000	0.003	0.000	44.337	44.369

Table 2.6: Average number of pulls from each resource in Setting 2

	Resource i					
	1	2	3	4	5	Total
Benchmark	15.374	19.424	15.155	15.349	15.839	81.141
Random	8.733	9.683	8.577	8.780	8.871	44.644
TS-Prophet	6.919	14.992	3.980	9.127	9.350	44.369

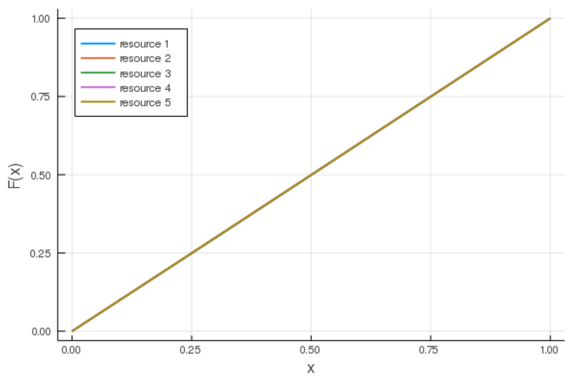


Figure 2-5: Function curves of $F_i(x)$ for $i \in [N]$ in Setting 1

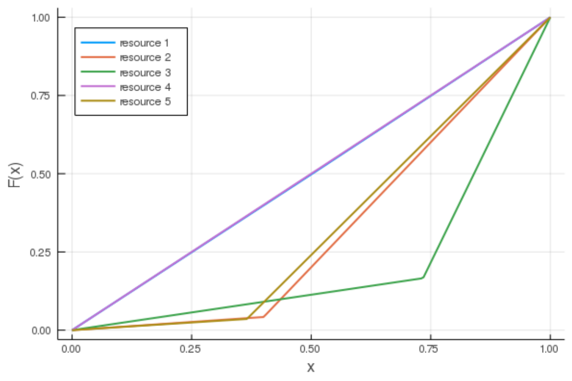


Figure 2-6: Function curves of $F_i(x)$ for $i \in [N]$ in Setting 2

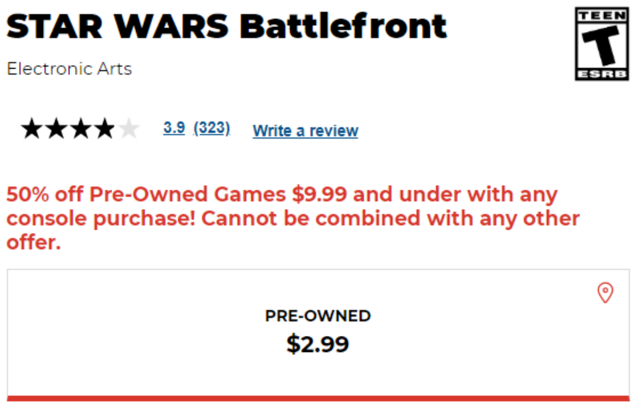
Chapter 3

Online Learning and Optimization for Add-on Discounts

3.1 Introduction

The video game industry has been growing fast and steadily in the past two decades. According to reports, in 2018, the U.S. video game industry matches that of the U.S. film industry on basis of revenue, making around 43 billion USD, and according to research by market analysts Newzoo, in 2018, the global games market value across all platforms is around 135 billion USD. The huge growth potential of the video game industry is also shown by the rapid sales increase during the coronavirus (COVID-19) pandemic. According to the weekly sales data from GSD, 4.3 million games are sold globally during the week of March 16, 2020, which amounts to a rise of 63% over the week prior.

Major platforms for video games include PCs, mobile phones, video game consoles and virtual reality (VR) headsets. Unlike PCs and mobile phones, video game consoles and VR headsets mainly support game functions. A unique structure for purchasing games for these devices is that customers have to first commit to the hardware, which is usually expensive, and then purchase the games, which are cheaper but include a large number of selections. For retailers, this unique structure motivates a creative *add-on discount strategy* for sales promotion, where a retailer offers customers discounts on a number of selected games after the customer makes a purchase of a video game console or a virtual reality headset. Figure 3-1 shows an example of this strategy from Gamestop Corp., a major game retailer in the U.S. In this example, customers enjoy discounts on certain types of games if they purchase the games together with a game console.


STAR WARS Battlefront

Electronic Arts

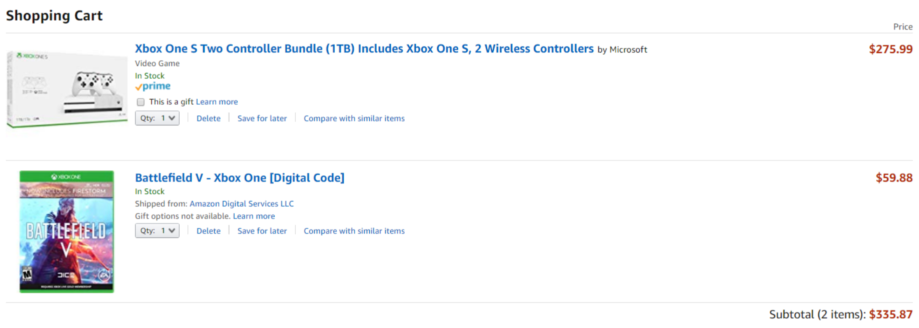
50% off Pre-Owned Games $9.99 and under with any console purchase! Cannot be combined with any other offer.

PRE-OWNED

$2.99

Figure 3-1: Add-on discount example from Gamestop Corp, retrieved on 2019-08-18.

The add-on discount strategy is different from product bundling. With add-on discounts, customers can make free selections from the offered set of add-on products, while with product bundling, customers can select only from fixed bundles of products. Moreover, although retailers can offer every possible product combination with add-on selection as a product bundle and decide each bundle's price individually, such a strategy is not efficient in practice, and more importantly, might cause price inconsistencies. Figure 3-2 shows an example of price inconsistency. In this example, consider a customer who wants to buy a game console, an extra controller and a certain game. If the customer chooses combination 1, which contains a game and a console-controller bundle, the final price would be \$335.87. However, if the customer chooses combination 2, which contains a controller and a console-game bundle, the final price would be \$281.98. This significant price difference for the same selection of products results in an inconsistent environment, which not only creates bad shopping experience for customers, but also financially damages the retailer's business in the long run. In contrast, for the add-on discount strategy, price inconsistency does not exist because the final price always equals the sum of the prices of the selected products, and is thus independent from the way the products are combined.



Shopping Cart

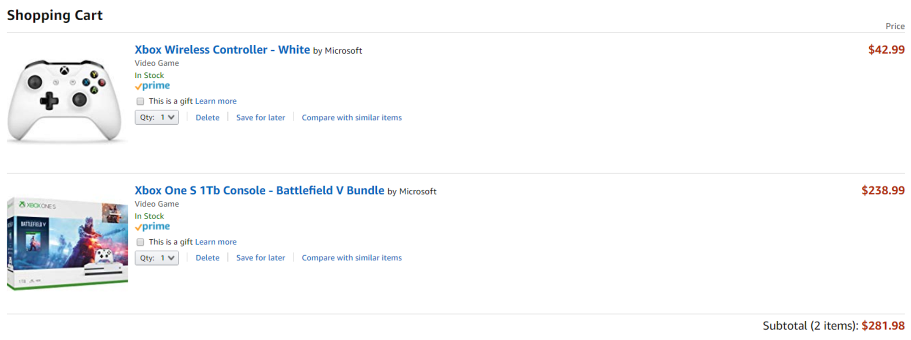
Price

Xbox One S Two Controller Bundle (1TB) Includes Xbox One S, 2 Wireless Controllers by Microsoft — $275.99
Video Game
In Stock
prime
This is a gift Learn more
Qty: 1 | Delete | Save for later | Compare with similar items

Battlefield V - Xbox One [Digital Code] — $59.88
In Stock
Shipped from: Amazon Digital Services LLC
Gift options not available. Learn more
Qty: 1 | Delete | Save for later | Compare with similar items

Subtotal (2 items): $335.87

(a) Combination 1.



Shopping Cart

Price

Xbox Wireless Controller - White by Microsoft — $42.99
Video Game
In Stock
prime
This is a gift Learn more
Qty: 1 | Delete | Save for later | Compare with similar items

Xbox One S 1Tb Console - Battlefield V Bundle by Microsoft — $238.99
Video Game
In Stock
prime
This is a gift Learn more
Qty: 1 | Delete | Save for later | Compare with similar items

Subtotal (2 items): $281.98

(b) Combination 2.

Figure 3-2: Example of price inconsistency for the same selection of products. Retrieved on 2019-10-23 from Amazon.com.

The add-on discount structure exploits the complementary effects between products, and thus adds new dimensions to the traditional pricing problem. Specifically, in addition to deciding the original prices of different products, retailers now also need to the decide the selection of add-on products, as well as their add-on discounts. From the basic principle of optimization, we know that adding new dimensions enlarges the feasible region of a problem, and hence leads to better decisions. Therefore, by using the add-on discount strategy, a retailer can expect a higher revenue than using the same pricing strategy with no add-on discounts.

Regardless of the advantages of the add-on discount strategy, retailers may be hesitant to implement the strategy in practice due to the following challenges. One of the challenges is to limit the number of add-on discounts. For example, when retailers show discount offers via pop-up messages on the customer's checkout page, there is usually a space limit on the total number of displayed offers. In addition, if a retailer offers too many add-on discounts, other retailers might take this as an arbitrage opportunity by purchasing the products with discounts and selling them elsewhere at the original prices. In these cases, retailers need to take *space constraints* into account, and the constraint increases the complexity of the problem. Another challenge that might hold retailers back is the lack of past experience or historical data. In the scenario where retailers have no knowledge of the demand information, blindly offering discounts with the add-on structure would harm the total revenue. Hence retailers need to implement a learning algorithm together with the add-on discount strategy to learn the unknown parameters on the fly, and such design of the learning algorithm also increases the complexity of the problem.

Our Contributions. In this paper, we study the revenue management problem with add-on discounts. To the best of our knowledge, this is the first paper that formally studies this problem. In particular, we consider a joint learning and optimization problem, where the retailer does not know the demand functions of different products *a priori*, and has to learn the information on the fly based on real-time observations of customers' purchases. Our formulation of the problem incorporates both the primary demand for products at their original prices and the add-on purchases for products with selected discounts. We also consider a space limit constraint on the total number of add-on discount offers.

Our contributions in this paper can be summarized as follows.

We formulate the revenue management problem with add-on discounts as an optimization problem with mixed binary decision variables. In the offline setting where the retailer has access to all the demand information, we develop an approximation algorithm that can solve the problem to any desired accuracy. We also show that the algorithm is a Fully Polynomial-Time Approximation Scheme (FPTAS).

In the online setting where the retailer has no knowledge of the demand information, we develop an efficient UCB-based learning algorithm that uses FPTAS optimization algorithm as a subroutine. We show that the learning algorithm outputs a policy that converges to the offline optimal policy with a fast rate. We also show that the convergence rate is tight up to a certain logarithmic term.

We conduct numerical experiments based on the real-world transaction data we collect from Tmall.com. Based on our numerical results, we observe that our UCB-based learning algorithm has a robust performance and fast convergence rate in various test scenarios. In addition, we observe that the learning algorithm can quickly outperform the optimal policy that does not use add-on discounts. These observations illustrate the efficiency of our learning algorithm, as well as the advantages of using the add-on discount strategy in practice.

3.1.1 Literature Review

To the best of our knowledge, the add-on discount strategy has not been formally studied in the the Operations Management (OM) literature, despite the fact that the strategy has been a common practice in the video game industry. Chen et al. (2019c) study a similar but different model. In that paper, the authors' focus is to figure out what products to recommend to a customer at the checkout stage, given the customer's primary purchase and each product's remaining inventory. We highlight the difference between that paper and our paper as follows: (1) In Chen et al. (2019c), the focus is on the checkout stage, and they assume that customers' primary purchases are exogenous and not affected by the decision-maker. In contrast, in our model, the retailer controls both the primary purchase and the add-on purchase. (2) In their model, they assumed that when making the add-on recommendation for a certain products, there are two possible strategies: one is at the original price, and the other one is at a certain pre-determined discount price. In our model, we are not restricted to two alternatives. (3) In their model, they consider a fixed starting inventory. In our model, we do not include inventory. (4) From the methodology perspective, they focus on a competitive ratio analysis, and we consider the regret minimization.

Our work is also related to different areas of the literature: assortment planning, product bundling, multi-armed bandit problems and UCB algorithms. Due to space limitation, we do not provide an exhaustive review of the literature and only provide a brief literature review as follows.

Assortment Planning. The assortment planning problem models a customer's choice over a set of different products and focuses on finding the profit-maximizing assortment subject to various resource and capacity constraints. The problem has been studied extensively in the revenue management literature. In particular, in the offline setting where the underlying choice models are known, Talluri Van Ryzin (2004) propose an efficient algorithm for the single-resource assortment problem. Gallego et al. (2004), Liu Van Ryzin (2008), and Zhang Adelman (2009) then extend the choice-based models to network revenue management problems. Other works that study assortment algorithms under cardinality constraints, personalized decisions and various choice models can also be found in Kök et al. (2008), Davis et al. (2013), Golrezaei et al. (2014), Cheung Simchi-Levi (2016), Feldman Topaloglu (2017) and the references therein.

Recent research on assortment planning problems also focuses on the online setting where the parameters of the underlying choice models, such as multinomial logit (MNL), are not known and need to be learned online. In this line of work, Rusmevichientong et al. (2010), Agrawal et al. (2016a), Agrawal et al. (2017), Agrawal et al. (2019), and Miao Chao (2017) study the problem where every customer follows the same choice model; Kallus Udell (2016), Cheung Simchi-Levi (2017b), Bernstein et al. (2018), Miao et al. (2019), and Miao Chao (2019) study the problem where each customer follows a personalized choice model.

Different from the assortment planning problems that mainly focus on how customers select one product from a set of alternatives, our model emphasizes customers' add-on purchase dynamics.

Product Bundling. Both the add-on discount strategy and the bundling strategy are motivated by the complementary effects between products. There exist various product bundling strategies in the literature, such as pure bundling in which the retailer sells different products in a comprehensive bundle for a fixed price (Bakos Brynjolfsson (1999)), mixed bundling in which the retailer offers all possible product bundles alongside individual products (Chu et al. (2011a)), and customized bundling in which the retailer allows the customer to choose a certain quantity of products from a large pool of products for a fixed price (Hitt Chen (2005) and Wu et al. (2008)). We refer the readers to some recent papers (Ma Simchi-Levi (2015), Abdallah et al. (2017) and Abdallah (2019)) for a more in-depth review of the bundling literature.

As mentioned in the example in Figure 2, add-on discounts and bundling are different. The add-on discount strategy facilitates the customer's decision process, because with add-on discounts, the final price is only dependent on the set of products to purchase, not on how the bundles are formed.

Multi-Armed Bandit (MAB) problems and UCB algorithms. The multi-armed bandit problem is a useful tool to study sequential decision-making problems under unknown rewards, and there exist a large number of papers studying this problem in the online learning literature. For a comprehensive review of the classic MAB algorithms and their performance analysis, see Bubeck et al. (2012) and Slivkins (2019).

One of the classic multi-armed bandit models is the stochastic bandit, where the reward for pulling each arm is assumed to be i.i.d. drawn from an unknown probability distribution. In the seminal paper Auer et al. (2002), the authors provide an algorithm that keeps updating the estimation of upper confidence bound (UCB) of each arm's mean reward, and show that such an algorithm can obtain an accumulative regret of $O(\sqrt{T\log T})$ in T rounds. The UCB-type algorithm is widely used in various bandit settings, such as linear bandits (Rusmevichientong Tsitsiklis (2010), Abbasi-Yadkori et al. (2011), Chu et al. (2011b)), combinatorial bandits (Cesa-Bianchi Lugosi (2012), Jin et al. (2019)), and bandits with resource constraints (Badanidiyuru et al. (2013), Agrawal Devanur (2016)).

In the OM literature, recent research papers have also been focusing on problems under uncertain environments and applying bandit algorithms or other learning algorithms to tackle the exploration-exploitation tradeoffs in learning tasks. This includes dynamic pricing problems (Besbes Zeevi (2009), Besbes Zeevi (2012), Wang et al. (2014), Besbes Zeevi (2015), Ferreira et al. (2018), Gao et al. (2018)) and inventory control problems with unknown demand distributions (Zhang et al. (2017), Zhang et al. (2019), Chen et al. (2019a), Yuan et al. (2019)), assortment optimization problems with unknown purchase probabilities (Cheung Simchi-Levi (2017a), Agrawal et al. (2019)), online matching and resource allocation problems with unknown reward distributions (Cheung et al. (2018)).

3.2 Model

We present in this section the formulation of the revenue management problem with add-on discounts.

Consider a retailer managing two types of products: core products (e.g., different variants of a video game console from the same brand) and supportive products (e.g., video games for the same brand of video game consoles). Let N denote the number of core products, indexed by $\{1, \ldots, N\}$, and M the number of supportive products, indexed by $\{1, \ldots, N\}$. For each core product $n = 1, \ldots, N$, we assume that its price p_n is selected from set $\Omega_c := \{q_c^1, q_c^2, ..., q_c^{\zeta(c)}\}$, and for each supportive product, $m = 1, \ldots, M$, we assume that its price p_{N+m} is selected from set $\Omega_s := \{q_s^1, q_s^2, ..., q_s^{\zeta(s)}\}$. Let the binary variable $I_{N+m} \in \{0, 1\}$ with $m = 1, \ldots, M$ denote whether or not we offer an add-on discount for supportive product m. Denote the add-on discount price for product m as p'_{N+m}, and we assume p'_{N+m} is selected from $\Omega_a := \{q_a^1, q_a^2, ..., q_a^{\zeta(a)}\}$. In addition, as we discuss in Section 3.1, the retailer cannot offer too many add-on discounts. Thus, we consider in our model an additional space constraint that limits the total number of add-on discounts to be within S, i.e., $\sum_{m=1}^{M} I_{N+m} \leq S$.

On the demand side, there exist two types of purchases, differentiated by whether or not a customer has already owned a core product before they arrive at the retailer's online store. Since we consider the core product as a prerequisite for using supportive products (e.g., video game console for video games), we assume that customers will not consider purchasing any supportive product without owning or purchasing a core product first. This condition results in two types of purchases: A) purchases from customers that do not own a core product, and B) purchases from customers that have already owned a core product. For type A purchases, customers will first purchase a core product, and then they may or may not continue to purchase supportive products with or without add-on discounts. For type B purchases, customers will only consider purchasing supportive products without any add-on discounts. The purchase dynamics are further illustrated in Figure 3-3. In addition, we partition the purchases into two categories: primary demand and add-on purchase, as indicated by the colors of the arrows in the figure.

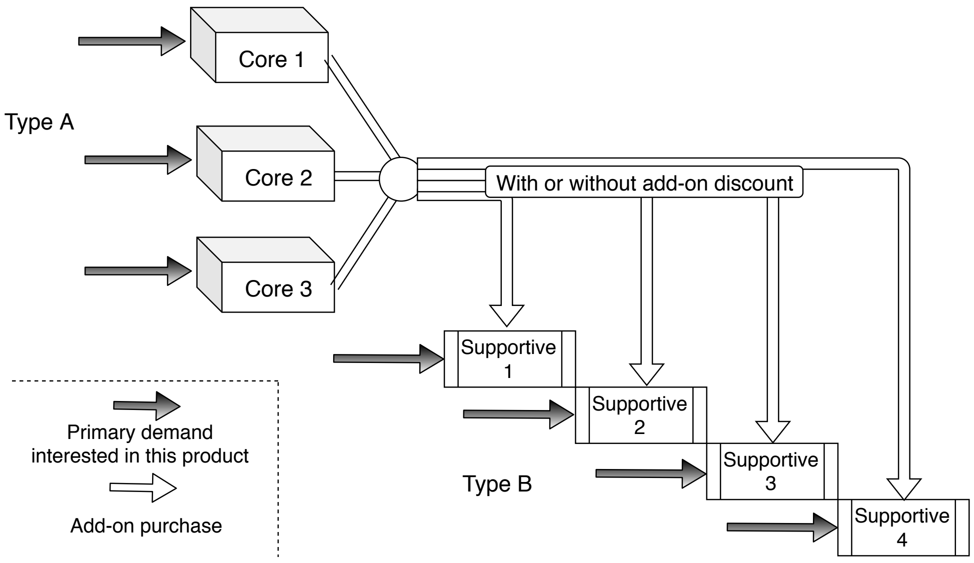


Figure 3-3: Illustration of a customer's purchase dynamics.

The entire selling horizon is divided into discrete time periods. We assume that each time period is short enough so that the primary demand for each core and each supportive product is a Bernoulli random variable. In particular, we use $\alpha_n(p_n)$ to denote the primary demand for core product $n = 1, \dots, N$, and $\alpha_{N+m}(p_{N+m})$ the primary demand for supportive product $m = 1, \dots, M$.

The add-on purchase category involves purchases both with and without discounts, depending on if we are offering add-on discount for each product, and we differentiate them as follows.

- Let $\beta'_{N+m}(p'_{N+m})$ be the probability that a customer continues to purchase product $N+m$ under discount price p'_{N+m}, after she purchases one core product.
- Let $\beta_{N+m}(p_{N+m})$ be the probability that a customer continues to purchase prod-

uct $N+m$ under original price p_{N+m} , after she purchases one core product.

We also assume that all demand parameters are independent across different products. We will discuss these model assumptions in detail in Section 3.5.

Let $\mathcal{R}$ denote the expected revenue per time period (each time period is identical). Given the retailer's goal of maximizing the total expected revenue, we can formulate the revenue management problem as:

$$\begin{aligned}
\max_{p_n, p_{N+m}, p'_{N+m}, I_{N+m}} \quad & \mathcal{R} := \sum_{n=1}^{N} \alpha_n(p_n)p_n + \sum_{m=1}^{M} \alpha_{N+m}(p_{N+m})p_{N+m} \\
& \quad + \left[\sum_{n=1}^{N} \alpha_n(p_n)\right] \cdot \sum_{m=1}^{M} \left[I_{N+m} \cdot \beta'_{N+m}(p'_{N+m}) \cdot p'_{N+m}\right] \\
& \quad + \left[\sum_{n=1}^{N} \alpha_n(p_n)\right] \cdot \sum_{m=1}^{M} \left[(1 - I_{N+m}) \cdot \beta_{N+m}(p_{N+m}) \cdot p_{N+m}\right] \\
\text{s.t.} \quad & \sum_{m=1}^{M} I_{N+m} \leq S, \\
& p'_{N+m} < p_{N+m} \text{ for } m = 1, \ldots, M, \\
& p_n \in \Omega_c \text{ for } n = 1, \ldots, N, \\
& p_{N+m} \in \Omega_s,\ p'_{N+m} \in \Omega_a,\ I_{N+m} \in \{0,1\} \text{ for } m = 1, \ldots, M.
\end{aligned} \tag{3.1}$$

In this optimization problem, the set of decisions include: the original price for each core product, the original and add-on discount price for each supportive product, and the binary indicator on whether or not to select each supportive product for add-on discount. The first term in $\mathcal{R}$ corresponds to the primary demand for core products, and the second term corresponds to the primary demand for supportive products. The third and fourth terms correspond to the add-on purchases for supportive products with and without add-on discounts, respectively. The first constraint sets the space constraint (upper bound) on the total number of add-on discounts. The second constraint requires that the discount price is less than the original price.

We observe from the formulation that the optimization problem is difficult to solve because the problem contains 1) discrete decision variables and 2) products of decision variables. In addition, the total number of feasible solutions is exponentially large, which makes the enumeration method intractable. Therefore, instead of finding the exact optimal solution, we propose in this paper an approximation algorithm that can solve the problem to any desired accuracy. We also show that the algorithm is an FPTAS, which means the running time of the algorithm is polynomial in both the problem size and the approximation error.

The optimization problem provides solutions to the revenue management problem in the offline setting where the demand functions $\alpha_n(\cdot)$, $\alpha_{N+m}(\cdot)$, $\beta_{N+m}(\cdot)$ and $\beta'_{N+m}(\cdot)$ are known. However, in practice, this information may not be available to the retailer due to the lack of historical transaction data, and the retailer then needs to learn the parameters online. In the following sections, we first present our solution to the offline optimization problem in Section 3.3. Then in Section 3.4, we propose a UCB-based learning algorithm that uses the offline optimization algorithm as a subroutine to solve the problem in the online setting.

3.3 Optimization Subroutine

In this section, we propose an approximation algorithm that can solve the offline optimization problem (3.1) to any desired accuracy, and show that the algorithm is an FPTAS.

As discussed in Section 3.2, the optimization problem is challenging due to the existence of discrete decision variables and products of decision variables. To resolve these challenges, we reformulate the original problem into two parts that separate the purchase of core products and the purchase of supportive products. We refer to these two problems as the *master* problem and the *subproblem*, respectively.

In the decomposed formulation, we replace the term $\sum_{n=1}^{N} \alpha_n(p_n)$ with γ, which represents the demand of core products per period. In addition, we introduce function $\mathcal{R}_s(\gamma)$ to denote the optimal revenue from the purchase of supportive products, which includes primary demand $\alpha_{N+m}(\cdot)$, add-on purchase $\beta_{N+m}(\cdot)$ and $\beta'_{N+m}(\cdot)$, when the demand for the core products is γ.

$$\textbf{Master problem.}\quad \begin{aligned} \max_{p_n} \quad & \mathcal{R} = \sum_{n=1}^{N} \alpha_n(p_n)p_n + \mathcal{R}_s(\gamma) \\ \text{s.t.} \quad & \sum_{n=1}^{N} \alpha_n(p_n) = \gamma \\ & p_n \in \Omega_c. \end{aligned} \tag{3.2}$$

$$\textbf{Subproblem.}\quad \mathcal{R}_s(\gamma) := \tag{3.3}$$

$$\begin{aligned} \max_{p_{N+m}, p'_{N+m}, I_{N+m}} \quad & \sum_{m=1}^{M} \alpha_{N+m}(p_{N+m})p_{N+m} \\ & + \gamma \cdot \sum_{m=1}^{M} \left[I_{N+m} \cdot \beta'_{N+m}(p'_{N+m}) \cdot p'_{N+m} \right] \\ & + \gamma \cdot \sum_{m=1}^{M} \left[(1 - I_{N+m}) \cdot \beta_{N+m}(p_{N+m}) \cdot p_{N+m} \right] \qquad (3.4) \\ \text{s.t.} \quad & \sum_{m=1}^{M} I_{N+m} \le S, \\ & p_{N+m} < p'_{N+m} \text{ for } m = 1, \dots, M \\ & p_{N+m} \in \Omega_s,\ p'_{N+m} \in \Omega_a,\ I_{N+m} \in \{0, 1\} \text{ for } m = 1, \dots, M. \end{aligned}$$

The decomposed formulation does not provide a tractable solution directly: in order to solve the problem, we need to determine the value of γ, which can take exponentially many values within $[0, N]$. Nevertheless, since γ is bounded, we can adopt a discretization approach that solves the problem for only a set of discrete points in $[0, N]$. In the following, we first show in Lemma 3.1 that function $\mathcal{R}_s(\gamma)$ is Lipschitz continuous. Then building on this lemma, we develop an approximation algorithm using the discretization approach to solve the master problem.

The high-level intuition for function $\mathcal{R}_s(\gamma)$'s Lipschitz continuity is based on the observation that parameter γ appears in the objective function of the subproblem. Thus, when the value of γ changes locally, the value of $\mathcal{R}_s(\gamma)$ should not change too much.

Lemma 3.1. *The function $\mathcal{R}_s(\gamma)$, as defined in* (3.3)*, is Lipschitz continuous in $\gamma \geq 0$ with parameter $M \cdot \hat{p}$, where $\hat{p}$ is the highest price among all the products, namely, $\hat{p} := \max_{p \in \Omega_c \cup \Omega_s} p$.*

Lemma 3.1 implies that we can approximate the value of $\mathcal{R}_s(\gamma)$ with a guaranteed accuracy. More importantly, this result motivates an approximation scheme where we only need to evaluate the value of $\mathcal{R}_s(\gamma)$ for a set of discrete points in $[0, N]$, instead of for all possible γ values.

Based on the approximation scheme, we can develop the solutions to the subproblem and the master problem separately. Specifically, for the subproblem, we can formulate it as a selection problem and solve it using a greedy approach. For the master problem, we can formulate it as a N-stage dynamic program, with approximations between stages, and solve it using backward induction.

We formally describe the detailed procedures of our algorithm in Algorithm 2. Then we show in Lemma 3.2 that Algorithm 2 has a polynomial runtime. Next, in Lemma 3.3, we show that Algorithm 2 has a bounded approximation error. Building on the results of these two lemmas, we show in Theorem 3.1 that Algorithm 2 is an FPTAS.

Lemma 3.2. *Algorithm 2 has a runtime of complexity $O(C^4 \cdot K)$, where*

$$C := \max(M, N, |\Omega_c|, |\Omega_s|, |\Omega_a|).$$

Proof. Consider the algorithm's runtime in the Big-O complexity. In Part 1 of Algorithm 2, step b) takes the longest runtime. Specifically, in step b), we enumerate $M \cdot N \cdot K$ cases in total, and solve each case by enumerating all possible pairs of p_{N+m} and p'_{N+m} such that $p_{N+m} > p'_{N+m}$. The runtime for step b) is thus $O\left(M \cdot N \cdot K \cdot |\Omega_s| \cdot |\Omega_a|\right) \leq O\left(C^4 \cdot K\right)$, and this also gives the runtime complexity of Part 1. In Part 2 of Algorithm 2, the total number of states is $O\left(C^2 \cdot K\right)$. In addition, for each state, we check the optimality equation once, which has runtime $O\left(C\right)$. The runtime complexity for Part 2 is thus $O\left(C^3 \cdot K\right)$. Combining the two parts, we obtain the algorithm's total runtime complexity $O\left(C^4 \cdot K\right)$.

Algorithm 1 FPTAS optimization subroutine for the offline optimization problem:

Algorithm input:

- Ω_c, Ω_s, Ω_a,
- $\alpha_n(p_n)$ for all $p_n \in \Omega_c$ and $n = 1, \ldots, N$,
- $\alpha_{N+m}(p_{N+m})$ for all $p_{N+m} \in \Omega_s$ and $m = 1, \ldots, M$,
- $\beta'_{N+m}(p'_{N+m})$ for all $p'_{N+m} \in \Omega_a$ and $m = 1, \ldots, M$,
- $\beta_{N+m}(p_{N+m})$ for all $p_{N+m} \in \Omega_s$ and $m = 1, \ldots, M$,
- Integer constant K.

Part 1: Solve supportive revenue part separately.

a) For all $m = 1, \ldots, M$, and $\gamma = 0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{NK}{K}$, solve

$$\max_{p_{N+m} \in \Omega_s} \alpha_{N+m}(p_{N+m}) p_{N+m} + \gamma \beta_{N+m}(p_{N+m}) p_{N+m},$$

and denote the optimal objective value as $r_{N+m}(\gamma)$.

b) For $m = 1, \ldots, M$, and $\gamma = 0, \frac{1}{K}, \frac{2}{K}, \ldots, \frac{NK}{K}$, solve

$$\max_{p_{N+m} \in \Omega_s, p'_{N+m} \in \Omega_a, p'_{N+m} < p_{N+m}} \alpha_{N+m}(p_{N+m}) p_{N+m} + \gamma \beta'_{N+m}(p'_{N+m}) p'_{N+m},$$

and denote the optimal objective value as $r'_{N+m}(\gamma)$.

c) For $\gamma = 0, 1/K, 2/K, \ldots, N$, sort the values of $r'_{N+m}(\gamma) - r_{N+m}(\gamma)$ into an array in the descending order. Set $I_{N+m}(\gamma) = 1$, if $r'_{N+m}(\gamma) - r_{N+m}(\gamma)$ is positive and in the first S entries of the array of sorted values. Set $I_{N+m}(\gamma) = 0$, otherwise.

d) For $\gamma = 0, 1/K, 2/K, \ldots, N$, let

$$\mathcal{R}_s(\gamma) = \sum_{m=1}^{M} \left[r_{N+m} + I_{N+m}(\gamma) \cdot \left[r'_{N+m}(\gamma) - r_{N+m}(\gamma) \right] \right].$$

Algorithm 2 FPTAS optimization subroutine for the offline optimization problem:

Part 2: Combining the revenue of core products and supportive products using dynamic programming.

a) *Initialization:* For $n = 1, \dots, N$ and $p_n \in \Omega_c$, let $\hat{\alpha}_n(p_n)$ be $\alpha_n(p_n)$ rounded to the nearest integer multiple of $1/K$.

b) *State:* (n, γ). *Action:* p_n in every state $(n, \cdot)$.

c) *Value function:* $V_n(\gamma)$ is defined as the maximum revenue to be earned from all the products (both core and supportive) excluding product 1 to $n-1$, when the total (approximate) demand for the first $n-1$ products are γ.

d) *Optimality equation:* $V_n(\gamma) = \max_{p_n \in \Omega_c} \left[\alpha_n(p_n) \cdot p_n + V_{n+1}(\gamma + \hat{\alpha}_n(p_n))\right].$

e) *Boundary condition:* $V_{N+1}(\gamma) = \mathcal{R}_s(\gamma)$, for all $\gamma = 0, 1/K, 2/K, \dots, N$.

f) The above DP can be solved efficiently using backward induction, the optimal decisions can be retrieved along the optimality equations, and $V_1(0)$ is the approximate optimal total revenue.

Lemma 3.3. *The approximation error of Algorithm 2 is upper-bounded by*

$$\frac{\hat{p}MN}{K},$$

where $\hat{p}$ is the highest price among all the products.

Theorem 3.1. *Suppose $V(OPT) \geq v^*$. For any problem instance and an $\varepsilon > 0$, Algorithm 2 can output an $(1-\varepsilon)$-optimal policy, with running time polynomial in both the problem size and $1/\varepsilon$, with parameter*

$$K = \left\lceil \frac{\hat{p}MN}{v^* \cdot \varepsilon} \right\rceil.$$

In other words, Algorithm 2 is an FPTAS.

Proof. By Lemma 3.3, we know that the approximation error of Algorithm 2 is bounded by $\frac{\hat{p}MN}{K}$. Given the value of K, we have

$$V(OPT) - V(ALG) \leq \frac{\hat{p}MN}{K} \leq \cdot \varepsilon v^*. \leq \ \varepsilon \cdot V(OPT).$$

Therefore, the algorithm is $(1-\varepsilon)$-optimal. By Lemma 3.2, we also know that the algorithm's runtime is polynomial in both the problem size and $1/\varepsilon$. Therefore, Algorithm 2 is an FPTAS.

3.4 Learning Algorithm and Regret Analysis

We consider in this section the revenue management problem in the online setting where the demand functions $\alpha_n(\cdot)$, $\alpha_{N+m}(\cdot)$, $\beta_{N+m}(\cdot)$ and $\beta'_{N+m}(\cdot)$ are not known *a priori*. In this setting, the retailer needs to determine the prices of different products and the selection of products with add-on discounts, while conducting price experiments and learning the demand information on the fly. More importantly, with the goal of maximizing the total revenue over T selling periods, the retailer faces the classic learning (exploration) and earning (exploitation) trade-off.

To tackle these challenges from unknown demand parameters, we model the joint learning and optimization problem as a multi-armed bandit, and develop a UCB-based algorithm to solve the problem. One way to design the algorithm is to construct the upper confidence bound (UCB) of the expected revenue (i.e., reward) of each policy (i.e., arm), which is equal to the empirical mean of each policy's revenue plus a confidence interval. Then the algorithm picks the policy with the highest upper confidence bound in each period. However, this naive construction of the UCBs results in the following issues.

- The learning algorithm is highly inefficient because the total number of policies in our problem is exponentially large. Consequently, the regret of this learning algorithm, as defined in (3.5), would be very large, meaning the algorithm can hardly converge to the optimal policy.
- In each period of the algorithm, it is impossible to compare an exponential number of policies so as to find the best one to implement. In addition, it is difficult to implement the learning algorithm together with the optimization subroutine we propose in Section 3.3.

To resolve these issues, we adopt an alternative way of constructing the UCBs: instead of estimating the UCBs for each policy, we estimate the UCBs for each unknown parameter, namely, $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$, $\beta'_{N+m}(p'_{N+m})$ and $\beta_{N+m}(p_{N+m})$, for $p_n \in \Omega_c$, $p_{N+m} \in \Omega_s$ and $p'_{N+m} \in \Omega_a$. Then, we can use these estimates as inputs to the FPTAS optimization subroutine to determine the "optimal" policy in each period.

3.4.1 The learning algorithm

In the UCB-based learning algorithm, we keep track of the empirical mean of demand parameters $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$, $\beta'_{N+m}(p'_{N+m})$, β_{N+m} for all products $n \in \{1, \ldots, N\}$, $m \in \{1, \ldots, M\}$ and all prices $p_n \in \Omega_c$, $p_{N+m} \in \Omega_s$, $p'_{N+m} \in \Omega_a$, respectively. We also keep track of the *counter* of each price, which counts the number of periods or the number of purchased products associated with the price, for each type of demand function.

We introduce the following notations in our algorithm.

- $\overline{\alpha}_n(p_n)$: the empirical average of $\alpha_n(p_n)$, for all $n = 1, ..., N$ and $p_n \in \Omega_c$.
- $\overline{\alpha}_{N+m}(p_{N+m})$: the empirical average of $\alpha_{N+m}(p_{N+m})$, for all $m = 1, ..., M$ and $p_{N+m} \in \Omega_s$.
- $\overline{\beta}'_{N+m}(p'_{N+m})$: the empirical average of $\beta'_{N+m}(p'_{N+m})$, for all $m = 1, ..., M$ and $p'_{N+m} \in \Omega_a$.
- $\overline{\beta}_{N+m}(p_{N+m})$: the empirical average of $\beta_{N+m}(p_{N+m})$, for all $m = 1, ..., M$ and $p_{N+m} \in \Omega_s$.
- $c_n(p_n)$: the number of periods that price p_n of core product n has been used, for all $n = 1, ..., N$ and $p_n \in \Omega_c$.
- $c_{N+m}(p_{N+m})$: the number of periods that price p_m of supportive product $N+m$ has been used, for all $m = 1, ..., M$ and $p_{N+m} \in \Omega_s$.
- $c^{(a,1)}_{N+m}(p'_{N+m})$: the number of core products purchased when product $N+m$ is selected as an add-on product under the discount price p'_{N+m}, for all $m = 1, ..., M$ and $p'_{N+m} \in \Omega_a$.
- $c^{(a,2)}_{N+m}(p_{N+m})$: the number of core products purchased when product $N+m$ is not selected as an add-on product but offered at the original price p_{N+m}, for all $m = 1, ..., M$ and $p_{N+m} \in \Omega_s$.

We also introduce the notion of *episode*, which is defined as a consecutive number of periods. In the algorithm, we update the “online" policy at the beginning of each episode, and then use the policy for a number of periods, until the episode terminates with certain *stopping rules*. Therefore, the length of each episode (in periods) is in fact a stopping time.

We refer to our learning algorithm as UCB-Add-On, and formally describe it in Algorithm 3.

In the beginning of each episode, the algorithm first uses the FPTAS optimization subroutine to solve an *optimistic* version of the problem, in which all the parameters are evaluated at their UCBs. In addition, given that the demand parameters are defined as Bernoulli random variables, we truncate all the UCBs at value 1.

We also observe that the parameter K increases as the number of episodes τ and time period t increase. By Theorem 3.1, we know the approximation error of the optimization subroutine decreases with τ, while the computation time increases. To mitigate the computational cost, we set the stopping criteria where given policy Π_τ, each episode ends when the value of at least one of the associated counters is doubled within the episode. By this construction, the length of each episode increases in τ. As a result, the algorithm calls the subroutine less frequently as time increases, and the output policy also becomes more stable.

3.4.2 Regret analysis

We analyze the performance of our learning algorithm by adopting the standard notion of regret. Let $\mathcal{R}^*$ be the one-period expected revenue of the optimal *clairvoyant* policy that has access to the full demand information, and $\mathcal{R}(\Pi_t)$ the expected revenue of the policy Π_t that is used by algorithm UCB-Add-On in period t. The regret of our algorithm is then defined as

$$Regret(T) := \mathbf{E}\left[\sum_{t=1}^{T} \mathcal{R}^* - \mathcal{R}(\Pi_t)\right]. \tag{3.5}$$

Algorithm 3 UCB-Add-On

```
• Initialization.
  Set all the empirical means and counters to 0. Set the value of $\varepsilon$.
• Loop. For each episode $\tau$,
    1. Let period $t$ denote the first period in $\tau$. Solve the optimization problem (3.1) using the FPTAS subroutine with the following inputs.
        * $\Omega_c$, $\Omega_s$, $\Omega_a$,
        * $\widetilde{\alpha}_n(p_n) = \min\left(1, \bar{\alpha}_n(p_n) + \sqrt{\frac{2\log t}{c_n(p_n)}}\right)$ for all $p_n \in \Omega_c$ and $n = 1, \ldots, N$
        * $\widetilde{\alpha}_{N+m}(p_{N+m}) := \min\left(1, \bar{\alpha}_n(p_{N+m}) + \sqrt{\frac{2\log t}{c_{N+m}(p_{N+m})}}\right)$ for all $p_{N+m} \in \Omega_s$ and $m = 1, \ldots, M$
        * $\widetilde{\beta}'_{N+m}(p'_{N+m}) := \min\left(1, \bar{\beta}'_{N+m}(p'_{N+m}) + \sqrt{\frac{2\log t}{c^{(a,1)}_{N+m}(p'_{N+m})}}\right)$ for all $p'_{N+m} \in \Omega_a$ and $m = 1, \ldots, M$
        * $\widetilde{\beta}_{N+m}(p_{N+m}) = \min\left(1, \bar{\beta}_{N+m}(p_{N+m}) + \sqrt{\frac{2\log t}{c^{(a,2)}_{N+m}(p_{N+m})}}\right)$ for all $p_{N+m} \in \Omega_s$ and $m = 1, \ldots, M$
        * $K = \left\lceil \frac{\sqrt{t}}{\varepsilon} \right\rceil$
    2. Denote the output policy as $\Pi_\tau$. Keep using policy $\Pi_\tau$ and updating the empirical means and counters in each period, accordingly.
    3. Terminate the episode when the value of at least one of the counters (associated with the selected add-on products and prices under policy $\Pi_\tau$) is doubled within the episode.
       Set $\tau = \tau + 1$.
```

Since $\mathcal{R}^*$ is an upper bound of $\mathcal{R}(\Pi)$ for any policy Π, the regret is always non-negative. In the following theorem, we state our main result on the upper bound of our learning algorithm's regret.

Theorem 3.2. *For any problem instance, the regret of algorithm UCB-Add-On can be upper-bounded by*

$$Regret\ (T) \leq \mathcal{O}\left(NM\hat{p}\left((1/\lambda) \cdot \sqrt{UT\log T} + \varepsilon\sqrt{T}\right)\right), \tag{3.6}$$

where $\hat{p}$ is the maximum price as defined in Lemma 3.1, λ is the lowest possible probability that the total primary demand is non-zero, $U := \max\{|\Omega_c|, |\Omega_s|, |\Omega_a|\}$ and ε is the input parameter to Algorithm 3.

Lower bound. We note that the regret bound (3.6) shown in Theorem 3.2 is tight up to the logarithmic term. More specifically, we can show that the regret is lower-bounded by $\Omega\left(NM\hat{p}\sqrt{UT}\right)$. The proof is based on constructing a problem instance where this part of the regret is inevitable for any learning algorithm.

Consider the instance where the add-on space limit is $S = 0$. In addition, the primary demand $\alpha_{N+m}(\cdot)$ are zero, and the add-on purchase probabilities $\beta_{N+m}(\cdot)$ are one, for all supportive products $m = \{1, \ldots, M\}$, and all prices $p_{N+m} \in \Omega_s$.

In this instance, the optimal policy is to set the prices of all the supportive products at the highest price. For simplicity, consider that there is only one price available for all supportive products, which is $\hat{p}$ as defined in Lemma 3.1.

Now we can translate the problem into a collection of independent MAB problems. In particular, for each core product, we obtain a regret lower bound $\Omega\left(M\hat{p}\sqrt{UT}\right)$. Summing up the regret of all N core products, we then obtain the regret lower bound $\Omega\left(NM\hat{p}\sqrt{UT}\right)$.

The proof of Theorem 3.2 is based on breaking down the total regret.

Before moving to the proof details, we first introduce the notations that we need in the analysis. Let $\overline{\alpha}_{n,t}(p_n)$, $\overline{\alpha}_{N+m,t}(p_{N+m})$, $\overline{\beta}'_{N+m,t}(p'_{N+m})$, $\overline{\beta}_{N+m,t}(p_{N+m})$, $c_{n,t}(p_n)$, $c_{N+m,t}(p_{N+m})$, $c^{(a,1)}_{N+m,t}(p'_{N+m})$ and $c^{(a,2)}_{N+m,t}(p_{N+m})$ be the values of the corresponding parameters at the beginning of period t, respectively. Then, with these notations, we define the collection of *events* $\mathcal{E}_t$, where each parameter's empirical mean at the beginning of period t is not in its confidence interval that is shown in Algorithm 3. Formally, we have

$$
\begin{aligned}
\mathcal{E}_t \ := \ & \left\{ \bigcup_{n\in[N]} \bigcup_{p_n\in\Omega_c} |\overline{\alpha}_{n,t}(p_n) - \alpha_n(p_n)| > \frac{2\log t}{c_{n,t}(p_n)} \right\} \\
& \bigcup \left\{ \bigcup_{m\in[M]} \bigcup_{p_{N+m}\in\Omega_s} |\overline{\alpha}_{N+m,t}(p_{N+m}) - \alpha_{N+m}(p_{N+m})| > \frac{2\log t}{c_{N+m,t}(p_{N+m})} \right\} \\
& \bigcup \left\{ \bigcup_{m\in[M]} \bigcup_{p'_{N+m}\in\Omega_a} \left|\overline{\beta}'_{N+m,t}(p'_{N+m}) - \beta'_{N+m,t}(p'_{N+m})\right| > \frac{2\log t}{c^{(a,1)}_{N+m,t}(p'_{N+m})} \right\} \\
& \bigcup \left\{ \bigcup_{m\in[M]} \bigcup_{p_{N+m}\in\Omega_s} |\overline{\beta}_{N+m,t}(p_{N+m}) - \beta_{N+m}(p_{N+m})| > \frac{2\log t}{c^{(a,2)}_{N+m,t}(p_{N+m})} \right\}.
\end{aligned}
$$

Conditioning on events $\mathcal{E}_{t(\tau)}$ for all episodes, we break down the total regret into two major parts. Let $t(\tau)$ denote the starting period of episode τ. Let $n(\tau)$ be the total number of episodes from period $t = 1$ to T, and $\ell(\tau)$ the length of episode τ, i.e., the total number of periods in episode τ. Specifically, we have

$$\begin{aligned}
\text{Regret}(T) &= \mathbf{E}\left[\sum_{t=1}^{T} \mathcal{R}^* - \mathcal{R}(\Pi_t)\right] \\
&= \mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} (\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau)\right] \\
&= \mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}_{t(\tau)}\right] \cdot \mathbf{P}\left[\mathcal{E}_{t(\tau)}\right]\right] \\
&+ \mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right] \cdot \mathbf{P}\left[\mathcal{E}'_{t(\tau)}\right]\right]. \qquad (3.7)
\end{aligned}$$

In the following, we bound the first term in (3.7) using Lemma 3.4 and Lemma 3.5, and bound the second term using Lemma 3.6. Theorem 3.2 then follows by summing up the two parts.

Lemma 3.4. *The expected length of the episode τ that starts with period t is upper-bounded by t, namely,*

$$\mathbf{E}[\ell(\tau)] \leq t(\tau).$$

Lemma 3.5. *Given the algorithm shown in Algorithm 3, we have*

$$\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}_{t(\tau)}\right] \cdot \mathbf{P}\left[\mathcal{E}_{t(\tau)}\right]\right] \leq K_1, \qquad (3.8)$$

where K_1 a constant that is independent of T.

Lemma 3.6. *Given the algorithm shown in Algorithm 3, we have*

$$\begin{aligned}
&\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right] \cdot \mathbf{P}\left[\mathcal{E}'_{t(\tau)}\right]\right] \qquad (3.9) \\
&\leq K_2 \cdot \left[NM\hat{p}\left((1/\lambda) \cdot \sqrt{UT\log T} + \varepsilon\sqrt{T}\right)\right], \qquad (3.10)
\end{aligned}$$

where K_2 is a constant that is independent of T.

The proof of Theorem 3.2 follows by combining the results in Lemma 3.4, 3.5, and 3.6.

3.5 Discussion

Given that the revenue management problem with add-on discounts is a new model, we discuss in this section several variants of the optimization problem for different practical scenarios. In particular, we discuss how the changes of the underlying model assumptions would affect the formulation of the problem, as well as the optimization and learning algorithms.

3.5.1 Assumption on independent demand

In optimization problem (3.1), we have assumed that all the demand parameters (both primary demand and add-on purchase) are independent across different products, which means that all the demand functions, $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$, $\beta'_{N+m}(p'_{N+m})$ and $\beta_{N+m}(p_{N+m})$, are dependent only on the price of the corresponding product itself. Alternatively, we can model demand using discrete choice models, e.g. the MNL model and other similar variants.

Consider the demand assumption that each customer can purchase at most one core product. Let $\mathbf{p}_c$ denote the vector of prices of the core products, i.e., $\mathbf{p}_c := (p_1, \dots, p_N)$, and $\alpha_n(\mathbf{p}_c)$ denote the purchase probability for core product n given price vector $\mathbf{p}_c$. Given a choice model, we have $\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c) \leq 1$. However, for supportive products, as one customer can purchase multiple supportive products at the same time, it is not appropriate to model demand functions $\alpha_{N+m}(\cdot)$, $\beta'_{N+m}(\cdot)$ and $\beta_{N+m}(\cdot)$ using choice models (because with choice models, e.g. MNL, each customer can select at most one product). The revenue management problem with a choice model for the core products can now be formulated as:

$$\begin{aligned}
\max_{p_n, p_{N+m}, p'_{N+m}, I_{N+m}} \quad & \sum_{n=1}^{N} \alpha_n(\mathbf{p}_c) \cdot p_n + \sum_{m=1}^{M} \alpha_{N+m}(p_{N+m}) \cdot p_{N+m} \\
& + \left[\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c)\right] \cdot \sum_{m=1}^{M} \left[I_{N+m} \cdot \beta'_{N+m}(p'_{N+m}) \cdot p'_{N+m}\right] \\
& + \left[\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c)\right] \cdot \sum_{m=1}^{M} \left[(1 - I_{N+m}) \cdot \beta_{N+m}(p_{N+m}) \cdot p_{N+m}\right] \\
\text{s.t.} \quad & \sum_{m=1}^{M} I_{N+m} \le S, \\
& p'_{N+m} < p_{N+m} \text{ for } m = 1, \dots, M, \\
& p_n \in \Omega_c \text{ for } n = 1, \dots, N, \\
& p_{N+m} \in \Omega_s,\ p'_{N+m} \in \Omega_a,\ I_{N+m} \in \{0, 1\} \text{ for } m = 1, \dots, M.
\end{aligned} \tag{3.11}$$

Given the new formulation (3.11), we cannot apply Algorithm 2 to solve the corresponding offline optimization problem. Although we can use Part 1 of Algorithm 2 to approximate the revenue function $\mathcal{R}_s(\cdot)$ for the supportive products, we cannot apply Part 2 of Algorithm 2 to solve the master problem $\max \sum_{n=1}^{N} \alpha_n(\mathbf{p}_c) p_n + \mathcal{R}_s(\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c))$ using the same dynamic programming approach. In the following, we discuss the solutions to the updated optimization problem in two cases: 1) when the number of core products N is small 2and) when N is large. We also discuss how to solve the joint learning and optimization problem in both cases.

When N is small relative to the computational resource, we can solve the optimization problem by enumerating the value of $\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c) p_n + \mathcal{R}_s(\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c))$ over all possible price vectors $\mathbf{p}_c$. Correspondingly, in the joint learning and optimization problem, we can consider the choice model as a black-box, and use this enumerating solution as a subroutine. Specifically, we can model each possible price vector $\mathbf{p}_c$ as an arm, and learn the values of $\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c) p_n$ and $\sum_{n=1}^{N} \alpha_n(\mathbf{p}_c)$ separately using a UCB-based algorithm similar to Algorithm 3. Moreover, we can show that such a learning algorithm can converge to the optimal policy, with a slower convergence rate, as the algorithm's regret is now proportional to the number of all possible price vectors, i.e., $O\left(|\Omega_c|^N\right)$.

When N is large and we can no longer consider the choice model as a black-box, we have to resort to heuristics based on neighborhood searching to obtain near-optimal solutions. In addition, we need to explicitly incorporate the choice model into the learning algorithm. In this case, if we assume the underlying choice model to be MNL, we can use the existing learning algorithms for MNL models (e.g., Rusmevichientong et al. (2010), Agrawal et al. (2016a), Agrawal et al. (2017), Agrawal et al. (2019), or Miao Chao (2017)) to handle the learning task in our problem.

3.5.2 Assumption on add-on demand function

We assume in the optimization problem (3.1) that the probability of an add-on purchase under discount price $\beta'_{N+m}(\cdot)$ depends only on discount price p'_{N+m}, rather than on both p'_{N+m} and p_{N+m}. This assumption is justified by the following two observations from practice. First, many supportive products, like video games, have *suggested retail prices* from the industry. For example, in the US, the prices of video games are regularly set at \$59.99. This fixed price, rather than the offered price p_{N+m}, can be considered as a reference point for customers. Therefore, it suffices to consider only the discount price p'_{N+m} in estimating demand $\beta'_{N+m}(\cdot)$. Second, in practice, add-on discounts are usually shown as a *limited time offer* as a way of triggering a customer's intention to purchase. Therefore, in the purchase dynamics, it is common that customers only consider whether or not to take discount price p'_{N+m}, instead of going back and comparing the discount price with the original price p_{N+m}.

Alternatively, one can assume that $\beta'_{N+m}(\cdot)$ depends both on the discount price p'_{N+m} and the original price p_{N+m}. In this case, we can update the formulation of the offline optimization problem as follows.

$$
\begin{aligned}
\max_{p_n, p_{N+m}, p'_{N+m}, I_{N+m}} \quad & \sum_{n=1}^{N} \alpha_n(p_n)p_n + \sum_{m=1}^{M} \alpha_{N+m}(p_{N+m})p_{N+m} \\
& \quad + \left[\sum_{n=1}^{N} \alpha_n(p_n)\right] \cdot \sum_{m=1}^{M} \left[I_{N+m} \cdot \beta'_{N+m}(p'_{N+m}, p_{N+m}) \cdot p'_{N+m}\right] \\
& \quad + \left[\sum_{n=1}^{N} \alpha_n(p_n)\right] \cdot \sum_{m=1}^{M} \left[(1 - I_{N+m}) \cdot \beta_{N+m}(p_{N+m}) \cdot p_{N+m}\right] \\
\text{s.t.} \quad & \sum_{m=1}^{M} I_{N+m} \leq S, \\
& p'_{N+m} < p_{N+m} \text{ for } m = 1, \ldots, M. \\
& p_n \in \Omega_c \text{ for } n = 1, \ldots, N, \\
& p_{N+m} \in \Omega_s, \; p'_{N+m} \in \Omega_a, \; I_{N+m} \in \{0, 1\} \text{ for } m = 1, \ldots, M.
\end{aligned}
\tag{3.12}
$$

In this updated formulation, we replace the $\beta'_{N+m}(p'_{N+m})$ in (3.1) with $\beta'_{N+m}(p'_{N+m}, p_{N+m})$. Note that this modification will not change the framework of the optimization subroutine shown in Algorithm 2, and hence we can still apply Algorithm 2 to solve the offline problem. More specifically, in Part 1.(b) of Algorithm 2, we simply update the procedure to

$$
\max_{p_{N+m} \in \Omega_s, p'_{N+m} \in \Omega_a, p'_{N+m} < p_{N+m}} \alpha_{N+m}(p_{N+m})p_{N+m} + \gamma \beta'_{N+m}(p'_{N+m})p'_{N+m},
$$

and the algorithm's complexity stays unchanged.

For the joint learning and optimization problem, we can adopt Algorithm 3 with a simple modification of the counters associated with function $\beta'_{N+m}(\cdot)$. Following a similar regret analysis procedure to that in Section 3.4, we obtain regret

$$
\mathcal{O}\left(NM\hat{p}\left(1/\lambda\sqrt{U^2 T \log T} + \varepsilon\sqrt{T}\right)\right),
$$

where the original term U shown in Theorem 3.2 is now replaced by U^2.

3.5.3 Assumption on Bernoulli demand

We assume in our model that all the demand parameters $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$, $\beta'_{N+m}(p'_{N+m})$ and $\beta_{N+m}(p_{N+m})$ are represented by Bernoulli random variables. In fact, the model can be extended to other types of demand parameters as long as we can obtain similar concentration results for the learning algorithm, as shown in Lemma 3.5 and Lemma 3.6. In our analysis, we use the Chernoff-Hoeffding inequality to obtain the concentration results for Bernoulli random variables. We refer interested readers to Bubeck et al. (2013) for further discussions on the concentration results for other types of random variables, such as normal, Poisson, exponential and all bounded distributions, which all belong to the family of sub-exponential distributions.

In this paper, our major focus is the add-on discount structure in the revenue management problem, and hence we skip the detailed discussion on the assumptions of the underlying demand random variables, as well as the corresponding regret analysis. In fact, relaxing the Bernoulli assumptions will only affect the construction of the UCB terms, and the framework of our learning and optimization algorithm will still apply. In addition, in practice, one may simply remove the $\min(1, \cdot)$ term in the UCB to handle other types of demand parameters.

3.6 Numerical Experiments

We present in this section the results of our numerical experiments. We conduct the experiments with the real-world data we collect from Tmall.com, which is an online e-commerce platform operated in China by Alibaba Group. The data provide the transaction history from a popular video gaming brand's official online store at Tmall.com. In the experiments, we first use the data to estimate the demand-price relationships of different products as the *ground truth*. Then we test the performance of algorithm UCB-Add-On in different settings with varying levels of add-on discount effects and add-on space limits. The experiment results not only validate the performance guarantee of the learning algorithm UCB-Add-On, as shown in Theorem 3.2, but also illustrate the advantages of using the add-on discount strategy in practice.

3.6.1 Experiment settings

The data provide the detailed transaction records from the video gaming brand's online store at Tmall.com during the period from October 2017 to July 2019. The store mainly sells video game consoles, video games and accessories. We observe from the data that the major sales are from three video game consoles and twenty video games. Therefore, we set $N = 3$ and $M = 20$ in all the experiments.

We use the data to calculate the *hourly* arrival rate, i.e, the number of customers per hour, as the demand for each of the selected video game consoles (core products) and video games (supportive products). In addition, for each product, given its demand under different prices, we use linear models to estimate the demand functions, i.e., $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$ and $\beta_{N+m}(p_{N+m})$, as the ground truth. More specifically, for each $m \in \{1, \ldots, M\}$, we estimate $\alpha_{N+m}(p_{N+m})$ and $\beta_{N+m}(p_{N+m})$ separately: if the game is purchased together with a game console, we then count the transaction as add-on demand $\beta_{N+m}(p_{N+m})$; and if the game is purchased without any game console, we count the transaction as primary demand $\alpha_{N+m}(p_{N+m})$. The details of the estimated coefficients of functions $\alpha_n(p_n)$, $\alpha_{N+m}(p_{N+m})$ and $\beta_{N+m}(p_{N+m})$ are provided in the Appendix. We note that the linear demand assumption is only used for estimating the ground truth, and is not known to the learning algorithm.

Since the online store does not implement any add-on discount strategy, we cannot estimate the add-on demand function $\beta'_{N+m}(p'_{N+m})$ from the transaction data directly. Instead, we generate these functions based on $\beta_{N+m}(p_{N+m})$ by making different assumptions about the level of the add-on discount effect. Given the intuition that for each video game, its add-on demand should be higher than its primary demand under the same price, we consider the following three cases in our experiments:

- **Low** add-on discount effect, where $\beta'_{N+m}(\cdot) = 2 \cdot \beta_{N+m}(\cdot)$ for all $m = 1, \ldots, M$;
- **Medium** add-on discount effect, where $\beta'_{N+m}(\cdot) = 3 \cdot \beta_{N+m}(\cdot)$ for all $m = 1, \ldots, M$;
- **High** add-on discount effect, where $\beta'_{N+m}(\cdot) = 4 \cdot \beta_{N+m}(\cdot)$ for all $m = 1, \ldots, M$.

Given the total number of games $M = 20$, we consider three possible values for the space limits, i.e., the total number of add-on discounts the retailer can offer at most, which is $S \in \{4, 6, 8\}$. In total, we test 9 cases (3 levels of add-on discount effect $\times$ 3 space limits) in our experiments.

We also modify the prices of different products that are used in practice. First, since the sale prices of video game consoles are much higher than those of the video games, we subtract the unit cost, which we assume to be $3,000$ CNY, from the sale price of each game console, in order to obtain prices of the same level in the objective function (3.1). With this modification, we can consider the objective value as the total profit rather than the total revenue. Second, for simplicity, we round the prices that end with 9 or 99 to the nearest ten or hundred. We set the price sets of different products as follows. For video game consoles, we have $p_n \in \Omega_c = \{200, 400, 600, 800\}$. For video games, we have $p_{N+m} \in \Omega_s = \{80, 100, 120, 140, 160\}$ and $p'_{N+m} \in \Omega_a = \{80, 100, 120, 140\}$. All the prices are in CNY. Note that price value 160 is removed from Ω_a, as it never makes a feasible add-on discount.

When running algorithm UCB-Add-On, we set the approximation error to be $\varepsilon = 0.1$ for the optimization subroutine. This approximation error is also used for calculating the revenue of the optimal policy with Algorithm 2. In addition, in constructing the confidence intervals, we add an additional multiplier, which we fix to be 2^{-3}, to all the UCB terms to enhance the algorithm's efficiency. The reasons for adding this multiplier are further discussed in Russo Van Roy (2014). Moreover, we note that each period in our experiment corresponds to one hour in the real world. This is consistent with our calculation of demand, which is defined as the hourly arrival rate. We also note that $365 \times 24 = 8760$ periods in our experiments correspond to the time of a year in the real-world.

3.6.2 Result Analysis

We aim to answer the following questions in analyzing the results of our numerical experiments.

- How does algorithm UCB-Add-On perform in different scenarios, in terms of the algorithm's rate of convergence to the optimal policy that knows the true demand functions?

- What is the optimality gap, namely, the difference in total revenue, between the optimal policy that uses add-on discounts (i.e., $S > 0$), and the optimal policy that does not use add-on discounts (i.e., $S = 0$), when both optimal policies know the true demand functions from the beginning?
- How long does it take for algorithm UCB-Add-On to achieve a better performance, in terms of total revenue (profit), than the optimal policy that does not use add-on discounts?

We summarize the experiment results under different test scenarios in Table 3.1.

In the first part of the table, we demonstrate the performance of algorithm UCB-Add-On. For each test scenario, we run the algorithm a total of 100 times, and then calculate the average regret, as defined in (3.5), up to period $T = 168$ (one week), period $T = 672$ (one month), period $T = 2016$ (three months) and period $T = 8760$ (one year), respectively. We display the average regret in percentage, which is given by

$$\frac{\text{Regret}(T)}{\mathcal{R}^* \cdot T} = 1 - \frac{\sum_{t=1}^{T} \mathbf{E}\left[\mathcal{R}(\Pi_t)\right]}{\mathcal{R}^* \cdot T}. \tag{3.13}$$

In the second part of Table 3.1, we answer the second question by displaying the difference of total revenue (in percentage) between the optimal policy that uses add-on discounts and the optimal policy that does not use add-on discounts. For simplicity, we call the first policy the *optimal (add-on) policy* and the second policy the *optimal no-add-on policy*. Let $\mathcal{R}_0^*$ be the revenue of the optimal no-add-on policy. The optimality gap percentage is given by $(\mathcal{R}^*/\mathcal{R}_0^* - 1)$.

In the last column of Table 3.1, we show the number of periods it takes for algorithm UCB-Add-On to surpass the revenue of the optimal no-add-on policy.

To visualize the performance of algorithm UCB-Add-On in comparison to the two optimal policies, we plot out the accumulative revenue of our algorithm as a function of the real-world time in Figure 3-4. Specifically, the results are from the test case where $S = 6$ and the add-on effect is medium. The plot also shows the comparisons between our algorithm and the other two optimal policies.

Table 3.1: Summary of experiment results under different test scenarios.

Low add-on discount effect : $\beta'_{N+m}(\cdot) = 2 \cdot \beta_{N+m}(\cdot)$						
	Expected average regret percentage				Optimality gap of	Time to beat optimal
Time	1 week	1 month	3 months	1 year	optimal no add-on policy	no add-on policy
$S = 4$	14.10%	10.60%	8.00%	5.50%	4.30%	*1.2 years*
$S = 6$	11.00%	9.20%	7.30%	3.90%	5.60%	*0.5 year*
$S = 8$	16.60%	11.40%	7.80%	5.10%	6.30%	*0.5 year*
Medium add-on discount effect: $\beta'_{N+m}(\cdot) = 3 \cdot \beta_{N+m}(\cdot)$						
	Expected average regret percentage				Optimality gap of	Time to beat optimal
Time	1 week	1 month	3 months	1 year	optimal no add-on policy	no add-on policy
$S = 4$	14.80%	11.30%	7.20%	5.30%	8.70%	*2 months*
$S = 6$	17.10%	11.90%	7.70%	4.80%	11.50%	*1 month*
$S = 8$	12.50%	10.10%	6.80%	4.50%	12.80%	*1/4 month*
High add-on discount effect: $\beta'_{N+m}(\cdot) = 4 \cdot \beta_{N+m}(\cdot)$						
	Expected average regret percentage				Optimality gap of	Time to beat optimal
Time	1 week	1 month	3 months	1 year	optimal no add-on policy	no add-on policy
$S = 4$	11.50%	8.30%	6.50%	4.50%	13.20%	*6 days*
$S = 6$	16.40%	11.20%	7.00%	4.40%	17.20%	*6 days*
$S = 8$	14.10%	9.30%	6.60%	4.10%	19.30%	*2 days*

First, we observe that algorithm UCB-Add-On can efficiently converge to the optimal policy in all test scenarios. The regret (in percentage) shrinks to 10% within one-month time in all the tests. In addition, the figure validates the algorithm's convergence rate $\mathcal{O}(\sqrt{T})$, as shown in Theorem 3.2.

Second, for the optimality gap between the two optimal policies, we observe that the gap increases when S becomes larger and when the add-on discount effect becomes stronger. The results are reasonable because in both cases, the revenue of the optimal add-on policy increases, while the revenue of the optimal no add-on policy stays the same. Moreover, we observe a non-negligible optimality gap: even in the modest setting where $S = 4$ and the add-on discount effect is low, the gap is 4.3%. Such comparison results demonstrate the advantages of using the add-on discount strategy.

Third, from the comparisons between algorithm UCB-Add-On and the optimal no-add-on policy, we observe that the time for algorithm UCB-Add-On to beat the optimal-add-on policy decreases as the space limit or the discount effect increases.

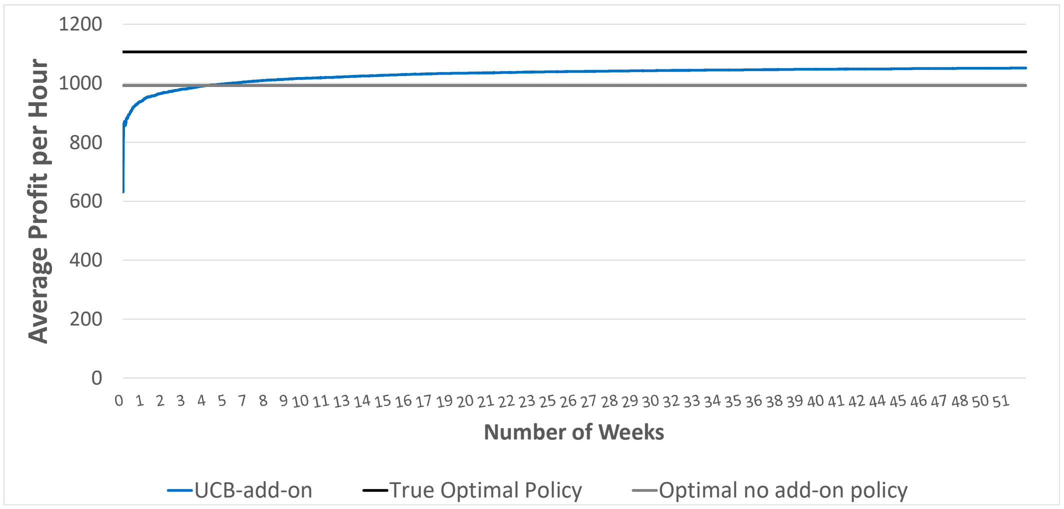


Figure 3-4: Performance of algorithm UCB-Add-On in the test case for $S = 6$ with medium add-on discount effect.

This is consistent with our observations from the optimality gap comparisons. More importantly, the results reassure the benefits of using the add-on discount strategy even when the retailer has no prior knowledge of all the demand parameters. As we show in Figure 3-4, where the x-axis depicts the real time in weeks, and the y-axis depicts the average hourly revenue (i.e., revenue per period), the learning algorithm can quickly outperform the optimal no-add-on policy in around four weeks.

Chapter 4

Fast Thompson Sampling for Online Network Revenue Management

4.1 Introduction

We study in this paper a canonical price-based network revenue management problem where a retailer needs to determine the prices of multiple products over a finite selling horizon. The products are made from multiple types of resources with limited inventory that are fixed before the selling season starts and cannot be replenished throughout the season. The retailer's goal is to maximize the total revenue by dynamically determining the prices of these products based on real-time observations of demand. The problem has a variety of applications including airline, hospitality, fashion retailing, and cloud computing. For example, in airlines, a firm needs to determine the prices of flight itineraries (products). Each flight consists of one or multiple flight legs (resources), and each flight leg has a finite number of seats (inventory). In cloud computing, a cloud service provider needs to determine the prices of virtual machines (products) that consume computing resources, including CPUs, storage, and memory (resources). Many other interesting applications of the problem can also be found in Talluri Van Ryzin (2006) and Özer et al. (2012).

One common challenge of using dynamic pricing in practice is that many retailers do not know the demand of each product associated with each price, and have to learn this information from the sales data during the selling horizon. In this case, the retailer faces the trade-off between trying different prices to learn and estimate the demand under each price (“exploration") and setting a price that maximizes the revenue based on the current demand estimations (“exploitation"). Specifically, given a finite selling horizon, if the retailer spends most of the selling season learning demand, the time left for exploiting this knowledge would be limited. On the other hand, if the retailer does not use enough periods for exploration, demand estimations would be inaccurate and thus yield sub-optimal pricing decisions. This exploration-exploitation trade-off is also signified by the resource inventory constraints.

One of the classic approaches for solving the exploration-exploitation trade-off in network revenue management problems with unknown demand is to use techniques from multi-armed bandits (MAB). In particular, by formulating the problem as an MAB, we can consider each possible selection of price vector as an “arm", and the revenue under the selected price vector as the “reward" of pulling an arm. The challenge of adopting this MAB approach is that in the presence of inventory constraints, traditional MAB algorithms, like Upper confidence bound (UCB) and Thompson sampling, are likely to fail in finding the optimal pricing policy. This is because these traditional algorithms are designed to find the price vector with the highest revenue, while the optimal pricing policy given inventory constraints should choose a combination of multiple price vectors over the entire selling season. To tackle this challenge, Ferreira et al. (2018) propose algorithms based on Thompson sampling techniques to solve the network revenue management problems with unknown demand. However, in their algorithms, the retailer needs to solve an estimated linear program (LP) in each period of the selling horizon. Given that solving LP is a time-consuming task, the algorithms in Ferreira et al. (2018) may not be a practical choice due to their computational drawbacks, especially for problems where the number of resources and products is huge.

We study in this paper a network revenue management problem with unknown demand (we refer to the problem as the "online network revenue management problem" as in Ferreira et al. (2018)), and we focus on developing *computationally efficient* dynamic pricing and learning algorithms that can maximize the total expected reward while learning the true demand parameter on the fly without violating the resource inventory constraints. Our main contributions can be summarized as follows.

We propose a primal-dual algorithm that does not require solving any LPs to solve the online network revenue management problem. In comparison to the algorithms in Ferreira et al. (2018), our algorithm has the advantage in computational efficiency, and more importantly, it does not compromise the convergence rate to the optimal dynamic pricing policy that knows the true demand parameter. Specifically, in Section 4.3.2, we show that our algorithm obtains the same order of regret as the algorithms in Ferreira et al. (2018). We also conduct numerical experiments and show in Section 4.3.3 that our algorithms outperforms the LP-based algorithms significantly in terms of computation time.

We build the connections between the online network revenue management problem and the multi-armed bandit problem from several aspects. We observe that our problem fits into the bandit with knapsack model proposed in Badanidiyuru et al. (2013) that study multi-armed bandit problems with resource constraints. If we model the network revenue management problem as a bandit, we have correlated “reward" and “resource consumption" for an “arm" since both items involve the mean demand under a selected price vector. This unique feature is useful in bounding the unit value of a resource, and thus motivates the design of our LP-free algorithms.

We show that our primal-dual decomposition framework provides not only the flexibility in designing algorithms, but also a convenient analysis framework. In particular, by using the primal-dual decomposition, we obtain a dynamic pricing and learning subproblem without any inventory constraints in the *primal* decision space. This subproblem is equivalent to a multi-armed bandit problem with no resource constraints, and we can then apply various bandit learning algorithms to learn the unknown demand parameters. Meanwhile, in the *dual* space, we obtain an online optimization subproblem that aims to learn the optimal resource value, and we can also apply various online convex optimization algorithms (see Hazan et al. (2016)) to solve the subproblem. We show in Section 4.4 several extensions of our model and algorithms to show the wide applicability of our primal-dual framework.

4.1.1 Literature Review

Due to challenge of incomplete demand information and the increased availability of real-time sales data, there has been a vast literature on dynamic pricing with demand learning, and most of the papers are based on the canonical revenue management problems defined in the seminal works Gallego Van Ryzin (1994) and Gallego Van Ryzin (1997). Specifically, Gallego Van Ryzin (1994) study a (single-leg) problem with a single product and a single resource, and Gallego Van Ryzin (1997) study a network revenue management problem with multiple products sharing multiple resources. Both papers are focused on dynamic pricing strategies under the assumption of complete demand information. For demand learning problems in the single-product single-resource setting, a series of algorithms with provably regret upper bounds are provided in Besbes Zeevi (2009) Lei et al. (2014) and Wang et al. (2014). A more detailed review of the problem can also be found in den Boer (2015). Our work focuses on demand learning in the network revenue management setting, which is more difficult to analyze than the single-leg problem. More discussions that compare these two problem settings can be found in Chen Shi (2019) and Chen et al. (2019b).

Several approaches have been proposed in the literature to address the exploration-exploitation trade-off in the general network revenue management problem with demand learning under resource constraints. One intuitive approach is to divide the selling horizon into a disjoint exploration phase and an exploitation phase (e.g., Besbes Zeevi (2012)). During the exploration phase, the retailer offers each price vector for a fixed number of times and collect the corresponding sales data. At the end of the phase, the retailer calculates the average sales as the estimation for the mean demand rate under each price, and then uses these estimates to find a combination of price vectors that maximize the revenue for the remaining periods, i.e., the exploitation phase, via a linear program. One drawback of this approach is that the demand learning process is discontinued in the exploitation phase, and thus the sales data collected in this later phase is wasted. Moreover, if the inventory is limited, it is very likely that retailer will run out of inventory in the exploration phase, and thus cannot exploit any learned demand information.

Another approach to address the exploration-exploitation trade-off is to use the multi-armed bandit tools. By formulating the learning task in a network revenue management problem as a multi-armed bandit, we can use algorithms like Upper Confidence Bound (UCB) and Thompson Sampling to estimate the demand parameters and use these estimates to optimize the prices in each discrete selling period. For example, by using UCB techniques, the algorithm creates a confidence interval for each unknown mean demand rate, and then optimize the prices based on the demand rates estimated at the upper bound of each interval. In addition, by using Thompson Sampling, the algorithm keeps updating the posterior distribution of each unknown mean demand rate, and optimize the prices in each time period based on the mean demand rates sampled from the latest posterior distribution. The original versions of these multi-armed bandit algorithms are developed without considering any resource constraints. Extensions of the algorithms in the presence of resource constraints can be found in Badanidiyuru et al. (2013) and Ferreira et al. (2018).

There also exist many other approaches for solving revenue management problems with unknown demand, such as the bisection search method in Wang et al. (2014) and Lei et al. (2014), the least square method in Besbes et al. (2015), and the primal dual method in Chen Gallego (2018). The methods in these papers heavily rely on the problem structure of a single product or a single resource, and can hardly be extended to a general multi-product multi-resource network revenue management problem.

We would also like to note that many papers reviewed above are primary focused on the discrete price setting, with an extension to the continuous price setting. One key distinction between discrete and continuous price sets in dynamic pricing and learning lies in the structure of their revenue optimization problems. As discussed in Ferreira et al. (2018), in the discrete price setting, the retailer needs to maximize over distributions of prices, since there might not exist a single price that maximizes revenue, while in the continuous price setting (e.g., prices drawn from a certain convex and compact set), there always exists a single price vector that is asymptotically optimal, if the demand function satisfies certain regularity conditions. In addition, as pointed out in Chen et al. (2019b), when analyzing the order of an algorithm's performance bound, the upper bound benchmarks used for the two settings are different (the continuous setting has a higher revenue upper bound). These distinctions make it unfair to compare the performance of the algorithms for different settings. More importantly, we cannot simply extend the existing algorithms developed in the discrete price setting to the continuous setting because of the difference in algorithm performance measures of the two settings. See more discussions in Chen et al. (2019b).

Our approach of solving the online network revenue management problem follows the multi-armed bandit approach with a computationally efficient primal-dual framework. When dealing with demand uncertainty, we closely follow the method in Ferreira et al. (2018) and use Thompson sampling to estimate the mean demand rates in each selling period. However, when optimizing the prices, instead of solving a linear program with sampled demand inputs and inventory constraints, we simply pick the price that maximizes the *pseudo* revenue, which is defined as the total estimated revenue (estimated demand times price for each product) minus the total value of resources that each product consumes. We measure the value of each resource by introducing a number of *dual* variables, each associated with a resource. At the end of each selling period, in addition to updating the posterior distribution of each unknown demand parameter, we also update the values of these dual variables based on the observed sales. Our primal-dual algorithm framework does not require solving any linear program, and therefore, can significantly outperform the algorithm in Ferreira et al. (2018) in terms of computational efficiency.

4.2 Fast Thompson Sampling

4.2.1 Model

We consider a retailer who sells N products, indexed by $i \in [N]$, over a finite selling horizon. Throughout the paper, we to $[x]$ to denote the set $\{1, 2, \ldots, x\}$. These products consume M resources, indexed by $j \in [M]$. Specifically, one unit of product i consumes a_{ij} units of resource j. The selling horizon is divided into T discrete selling periods. For each resource $j \in [M]$, it has I_j units of initial inventory, and no replenishment during the selling season. We use $I_j(t)$ to denote the inventory of resource j at the end of period $t \in [T]$, and we denote $I_j(0) = I_j$.

The sales dynamics in each period is as follows. At the beginning of period $t \in [T]$, the retailer offers a price for each product from a finite and discrete set of price vectors. We denote the set by $\{p_1, \ldots, p_K\}$, where for each $k \in [K]$, $p_k = (p_{1k}, \ldots, p_{Nk})$ is a vector of length N specifying the price of each product. We also assume there exists a "shut-off" price p_∞ such that the demand for any product under this price is zero with probability one. We use $P(t) = (P_1(t), \ldots, P_N(t))$ to denote the prices chosen by the retailer in this period, and we know $P(t) \in \{p_1, \ldots, p_K, p_\infty\}$.

Customers then observe the prices chosen by the retailer and make purchase decisions. We denote the demand of each product at period t by $D(t) = (D_1(t), \ldots, D_N(t))$. Given $P(t) = p_k$, we assume that demand $D(t)$ is sampled from a probability distribution with joint cumulative distribution function (CDF) $F(x_1, \ldots, x_N; p_k, \theta)$ that contains parameter θ, and we assume θ takes values in the parameter space Θ. The distribution is assumed to be sub-exponential. Note that many commonly used demand distributions belongs to the family of sub-exponential distributions, such as, normal, Poisson, exponential and all bounded distributions. We also assume that $D(t)$ is independent of the history $\mathcal{H}_{t-1} = \{P(1), D(1), \ldots, P(t-1), D(t-1)\}$ given $P(t)$.

If the inventory is sufficient to satisfy demand $D_i(t)$ for all products $i \in [N]$, the retailer receives revenue $\sum_{i=1}^N D_i(t)P_i(t)$, and the inventory of each resource $j \in [M]$ diminishes by $\sum_{i=1}^N D_i(t)a_{ij}$ units such that $I_j(t) = I_j(t-1) - \sum_{i=1}^N D_i(t)a_{ij}$.

Otherwise, if there exists at least one resource with insufficient inventory to satisfy the corresponding demand, the sales dynamics stops immediately. The retailer receives no revenue from this period, and the price is forced to be p_∞ in the remaining selling periods.

We assume that the demand parameter θ is fixed but unknown to the retailer at the beginning of the selling season, and the retailer needs to learn the true value of θ from the historical sales data $\mathcal{H}_{t-1}$. In addition, we assume the retailer knows a prior probability distribution of $\theta \in \Theta$ at the beginning. The retailer's objective is to maximize the expected total revenue over the entire selling horizon.

4.2.2 Relation to the Multi-Armed Bandit Problem

The model we describe is a generalization of the MAB problem, where we can consider each price vector as an "arm" and the associated revenue as the "reward". One deviation from the classic MAB model is that we consider inventory constraints in our problem. Note that in the presence of inventory constraints, the optimal pricing strategy should converge to a mixed strategy given by a distribution of multiple price vectors instead of a single price vector. Therefore, in addition to learning the unknown demand parameter, a good algorithm also needs to estimate the time when the inventory runs out. This is in contrast to classical MAB problems for which the process always ends at a fixed period.

Our model closely follows the dynamic pricing model in Besbes Zeevi (2012), Ferreira et al. (2018), and is closely related to the the bandit with knapsack (BwK) model studied in Badanidiyuru et al. (2013). In comparison to the BwK model, in this paper, we focus on the Bayesian setting where the prior distributions of the unknown parameters are known. Moreover, if we model the network revenue management problems with unknown demand as a BwK model, the "reward" and "consumption" of an "arm" are correlated since they both involves the mean demand parameters. This special feature provides convenience in designing computationally efficient algorithms. In Section 4.2.4 and Section 4.2.5, we will discuss the difference between our algorithm and other algorithms in the literature. We will explain the computational disadvantage of the LP-based learning algorithms in Ferreira et al. (2018) and the limitations of the primal-dual algorithm framework in Badanidiyuru et al. (2013). We also conduct numerical experiments to compare the performance of different algorithms and will present the results in Section 4.3.3.

4.2.3 Algorithm

We present our computational efficient Thompson Sampling algorithm Fast-TS in Algorithm 4. The algorithm is based on a primal-dual framework: in the primal space, we optimize price decisions based on sampled demand parameters, and in the dual space, we update a vector of *dual* variables λ_j, each associated with a resource inventory constraint I_j for $j \in [M]$. Intuitively, these dual variables measure the value, i.e., opportunity cost, of each unit of the corresponding resource.

In the initialization step of the algorithm, we define the domain of the dual vector by

$$\Omega = \{\lambda \mid \lambda \geq 0,\ \|\lambda\|_1 \leq \Lambda\}, \tag{4.1}$$

which requires each dual variable to be non-negative and the dual vector's 1-norm to be bounded by a constant Λ. When the selling horizon begins, in Step 1, we first use the Thompson sampling method to estimate the mean demand for each price vector. Then in Step 2, we select the price vector that maximizes the estimated mean *pseudo* revenue where the *price* of each product is subtracted by the estimated *cost* of the associated resources. After observing the realized demand, we update the posterior distribution of the demand parameters and the values of the dual variables respectively in Step 3 and Step 4. The update of posterior distribution follows that of the Thompson sampling algorithm, and the dual variables are updated by the online gradient descent method that is described in Algorithm 5 given objective function $g_t(\lambda)$ and step size η_t. The algorithm stops when at least one of the resources is out of inventory.

Algorithm 4 Fast Thompson Sampling (Fast-TS)

0. Initialization. Define *dual* domain Ω. Set $\lambda_j(0) = 0$ and $I_j(0) = I_j$ for all $j \in [M]$.

For each time period $t = 1, \ldots, T$:

1. Sample demand. Sample $\theta(t)$ from Θ according to the posterior distribution of $\theta(t)$ given history $\mathcal{H}_{t-1}$. Let $d_{ik}(t)$ be the mean demand given $\theta(t)$.

2. Optimize pricing decisions. Offer the optimal price vector with index $k(t)$ such that

$$k(t) = \arg\max_{k \in [K+1]} \sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j(t) a_{ij} \right) d_{ik}(t). \tag{4.2}$$

3. Update estimate of parameter θ. Observe demand $D(t)$. Update the history $\mathcal{H}_t = \mathcal{H}_{t-1}$ and the posterior distribution of θ given $\mathcal{H}_t$.

4. Update parameter λ. Define function

$$g_t(\lambda) = \sum_{j=1}^{M} \lambda_j \cdot (I_j/T - \sum_{i=1}^{N} a_{ij} D(t)). \tag{4.3}$$

Update $\lambda(t+1)$ by Online Gradient Descent with step size $\eta_t = C/\sqrt{t}$.

5. Stopping condition. Update the inventory of each resource $j \in [M]$. If there exists j such that inventory $I_j(t) \leq 0$, then Exit.

Algorithm 5 Online Gradient Descent

Initialization. Set $\eta_t = C/\sqrt{t}$. Calculate $\bigtriangledown g_t(\lambda(t))$ given $g_t(\lambda)$.

Update. Calculate $\lambda' = \lambda(t) - \eta_t \cdot \bigtriangledown g_t(\lambda(t))$.

Projection. Calculate $\lambda(t+1) = \Pi_\Omega(\lambda')$.

The update of the dual variables in Algorithm 5 is based on the online gradient descent (OGD) algorithm for an online convex optimization (OCO) problem. Specifically, the OCO problem is

$$\min_{\lambda \in \Omega} \sum_{t=1}^{T} \sum_{j=1}^{M} \lambda_j \left(\frac{I_j}{T} - \sum_{i=1}^{N} a_{ij} D(t) \right). \tag{4.4}$$

We will later see in the analysis (Section 4.3.2) that the optimal solution to this OCO problem actually corresponds to the optimal value of opportunity cost associated with each resource inventory constraint. Let $I_{\min}$ be the minimum inventory, namely, $I_{\min} = \min_{j\in[M]} I_j$, and $I_{\max}$ the maximum inventory, namely, $I_{\max} = \max_{j\in[M]} I_j$. In addition, let $p^j_{\max}$ be the maximum revenue that can possibly be achieved by adding one unit of resource j, namely, $p^j_{\max} = \max_{i:a_{ij}\neq 0, k\in[K]} (p_{ik}/a_{ij})$. We set the 1-norm upper bound of dual vector λ in domain Ω as

$$\Lambda = (I_{\max}/I_{\min}) \cdot \sum_{j=1}^{M} p^j_{\max}. \tag{4.5}$$

For step size $\eta_t = C/\sqrt{t}$, we have $C = D/G$ where D is the diameter of domain Ω, and G is the upper bound of the norm of gradient $\| \bigtriangledown g_t(\lambda)\|$. We set $D = \sqrt{2}\,\Lambda$, and will specify the value of G later in Section 4.3.2.

We note that the online gradient descent algorithm involves a projection step. Given λ', the projection to domain Ω is mathematically defined as

$$\Pi_\Omega(\lambda') = \arg\min_{\lambda\in\Omega} \|\lambda - \lambda'\|. \tag{4.6}$$

This projection result can be efficiently calculated without solving any LP. See Duchi et al. (2008). We provide the detail of the projection algorithm in the Appendix.

4.2.4 Comparison to LP-based Thompson Sampling

Ferreira et al. (2018) provide two LP-based Thompson sampling algorithms, denoted by TS-Fixed and TS-Update, to solve the online network revenue management problem. Those algorithms also use Thompson sampling as the learning backbone, as in Step 1 and Step 3 of our algorithm Fast-TS. The difference is that in Step 2, both TS-Fixed and TS-Update need to solve the following LP that uses the sampled mean demand $d(t)$ as inputs to optimize the price decisions. Specifically, algorithm TS-Fixed uses fixed inventory constraints $c_j = I_j/T$ and algorithm TS-Update uses updated inventory constraints that replace c_j with $c_j(t) = I_j(t-1)/(T-t+1)$.

$$\mathsf{LP}(t): \max_{x} \sum_{k=1}^{K} \left(\sum_{i=1}^{N} p_{ik} d_{ik}(t) \right) x_k \tag{4.7}$$

$$\text{s.t.} \sum_{k=1}^{K} \left(\sum_{i=1}^{N} a_{ij} d_{ik}(t) \right) x_k \leq c_j, \ \forall j \in [M] \tag{4.8}$$

$$\sum_{k=1}^{K} x_k \leq 1, \tag{4.9}$$

$$x_k \geq 0, \ \forall k \in [K]. \tag{4.10}$$

In comparison to algorithms TS-Fixed and TS-Update, our algorithm Fast-TS features a primal-dual framework that does not require solving any LP. Specifically, our algorithm uses dual variables to measure the tightness of the inventory constraints. These dual values correspond to the value (i.e. opportunity cost) of each unit of the resources, and are thus taken into account in the price optimization step (Step 2) of Fast-TS. By Lagrangian relaxation, we know the price decisions given by our algorithm can approximate the solution of the LP in TS-Fixed when these dual variables converge to their optimal values. Therefore, we can show that our algorithm has a similar performance guarantee as TS-Fixed. Moreover, since our algorithm is LP-free, it has the advantage in computations, and can thus outperform TS-Fixed significantly in terms of computational efficiency, especially for large-scale systems for which solving an LP in each iteration is very time expensive.

4.2.5 Comparison to primal-dual bandit with knapsack

By formulating the online network revenue management problem as a multi-armed bandit problem with resource constraints, we can also apply the primal-dual algorithm PD-BwK that is proposed in Badanidiyuru et al. (2013) to solve our problem. The algorithm PD-BwK uses UCB as the learning backbone, and is designed for problems with no prior knowledge. In each iteration, it estimates the upper bounds on revenue, lower bounds on resource consumption, and the dual price of each resource, and then selects the price vector that has the highest revenue-to-resource-price ("bang-perbuck") ratio.

Although PD-BwK is also based on an LP-free primal-dual framework, the algorithm has limitations compared to our algorithm Fast-TS. First, the original algorithm assumes that the revenue and resource consumption under each price vector is bounded by $[0,1]$ and $[0,1]^M$, and it uses the multiplicative weights update method to learn the optimal dual values. However, in the case where these bounds do not hold (e.g. Poisson demand process), we cannot apply the algorithm directly because its dual update method fails to specify the range of the optimal dual solution correctly. Second, PD-BwK is not as flexible as Fast-TS. In particular, algorithm PD-BwK and its analysis can hardly be adopted to the contextual setting, while our algorithm's primal-dual framework can be easily extended. More details about the extensions of our algorithm are provided in Section 4.4.

4.2.6 Examples

We use the following two examples to elaborate the update process of the Thompson sampling algorithm in Step 1 and Step 3 of algorithm Fast-TS that is provided in Algorithm 4.

Example 1. Bernoulli demand with independent Beta prior. We assume the demand for each product and all prices is Bernoulli distributed, and the unknown parameter θ denotes the mean demand rate $d(\theta)$. We also assume that θ has a Beta prior distribution, which is conjugated to the Bernoulli distribution. More specifically, we assume the prior distribution of mean demand $d_{ik}(\theta)$ is uniform in $[0,1]$, which is equivalent to a Beta$(1,1)$ distribution, and is independent for all $i \in [N]$ and $k \in [K]$.

The update of posterior distribution is easy to calculate. Let $N_k(t-1)$ be the number of periods that the retailer uses price p_k in the first $t-1$ periods, and $W_{ik}(t-1)$ the number of periods that product i is purchased under price p_k. In step 1 of the algorithm, we sample $d_{ik}(\theta)$ independently from the posterior distribution Beta$(W_{ik}(t-1)+1, N_k(t-1) - W_{ik}(t-1)+1)$ for each product i and price vector k. In step 3, given selected price vector $k(t)$ and observed demand $D_i(t)$ for product i, we update the posterior distribution of θ by updating the parameters

$N_{k(t)}(t) \leftarrow N_{k(t)}(t-1)+1$ and $W_{ik(t)}(t) \leftarrow W_{ik(t)}(t-1)+D_i(t)$ for all $i \in [N]$. The posterior distribution associated with the unchosen price vectors $k \neq k(t)$ are not changed.

Example 2. Poisson demand with independent Gamma prior. We assume the demand for each product and all prices is Poisson distributed, and the unknown parameter θ denotes the mean demand rate $d(\theta)$. We also assume that θ has a Gamma prior distribution, which is conjugated to the Poisson distribution. More specifically, we assume the prior distribution of mean demand $d_{ik}(\theta)$ is exponential with CDF $f(x) = e^{-x}$, which is equivalent to a Gamma$(1,1)$ distribution, and is independent for all $i \in [N]$ and $k \in [K]$.

The update of posterior distribution is also easy to calculate. Let $N_k(t-1)$ be the number of periods that the retailer uses price p_k in the first $t-1$ periods, and $W_{ik}(t-1)$ the total demand of product i under price p_k. In step 1 of the algorithm, we sample $d_{ik}(\theta)$ independently from the posterior distribution Gamma$(W_{ik}(t-1)+1, N_k(t-1)+1)$ for each product i and price vector k. In step 3, given selected price vector $k(t)$ and observed demand $D_i(t)$ for product i, we update the posterior distribution of θ by updating the parameters $N_{k(t)}(t) \leftarrow N_{k(t)}(t-1)+1$ and $W_{ik(t)}(t) \leftarrow W_{ik(t)}(t-1)+D_i(t)$ for all $i \in [N]$. The posterior distribution associated with the unchosen price vectors $k \neq k(t)$ are not changed.

4.3 Performance Analysis

In this section, we are focused on answering the following two questions. First, what is the performance guarantee of our algorithm in terms of its convergence rate to the optimal pricing policy that knows the true demand parameters? Second, what is our algorithm's computational efficiency compared to other algorithms in the literature? We provide in Section 4.3.2 and Section 4.3.3, respectively, answers to these two questions. In addition, building on the numerical results, we will discuss the trade-off between an algorithm's performance guarantee and its computational efficiency in Section 4.3.3.

4.3.1 Benchmark

We use *regret* to evaluate an algorithm's performance guarantee, which is defined as the revenue loss of an algorithm relative to the (benchmark) optimal pricing policy of a clairvoyant that knows the true demand parameter. Specifically, given true demand parameter θ, we define regret the in T selling periods by

$$\text{Regret}(T, \theta) = \mathbb{E}[\text{Rev}^*(T) \mid \theta] - \mathbb{E}[\text{Rev}(T) \mid \theta] \tag{4.11}$$

where $\text{Rev}^*(T)$ is the revenue achieved by the optimal policy of a clairvoyant, and $\text{Rev}(T)$ is the revenue achieved by an algorithm that does not know θ. The conditional expectation is taken over random demand realizations given θ, and any randomization within the algorithm.

We also define the Bayesian regret by

$$\text{BayesRegret}(T) = \mathbb{E}[\text{Regret}(T, \theta)], \tag{4.12}$$

where the expectation is taken over the prior distribution of θ.

The revenue of the optimal policy with known demand parameter can be found by dynamic programming. However, this approach is computationally intractable due to the curse of dimensionality. Gallego Van Ryzin (1997) have shown that the optimal revenue can be upper-bounded by the following linear program. Given true demand parameter θ, we denote the LP by $\mathsf{LP}(\theta)$, and the mean demand by $d(\theta)$.

$$\mathsf{LP}(\theta): \max_x \sum_{k=1}^{K} \left(\sum_{i=1}^{N} p_{ik} d_{ik}(\theta) \right) x_k \tag{4.13}$$

$$\text{s.t.} \sum_{k=1}^{K} \left(\sum_{i=1}^{N} a_{ij} d_{ik}(\theta) \right) x_k \leq \frac{I_j}{T}, \ \forall j \in [M] \tag{4.14}$$

$$\sum_{k=1}^{K} x_k \leq 1, \tag{4.15}$$

$$x_k \geq 0, \ \forall k \in [K]. \tag{4.16}$$

Let $\mathsf{OPT}(\theta)$ be the optimal value of $\mathsf{LP}(\theta)$. By the result shown in Gallego Van Ryzin (1997), we have

$$\mathbb{E}[\mathrm{Rev}^*(T) \mid \theta] \le \mathsf{OPT}(\theta) \cdot T. \tag{4.17}$$

Therefore, we have

$$\mathrm{Regret}(T, \theta) \le \mathsf{OPT}(\theta) \cdot T - \mathbb{E}[\mathrm{Rev}(T) \mid \theta], \tag{4.18}$$

and

$$\mathrm{BayesRegret}(T) \le \mathbb{E}[\mathsf{OPT}(\theta)] \cdot T - \mathbb{E}[\mathrm{Rev}(T)]. \tag{4.19}$$

4.3.2 Analysis of Fast Thompson Sampling

We provide the Bayesian regret bound of FastTS in Theorem. 4.1. In the analysis, we assume that the observed demand for each product $i \in [N]$ under any price vector p_k for $k \in [K]$ is bounded, namely, $D_i(t) \in [0, \bar{d}_i]$. The regret result involves the following constants:

$$q_{\max} = \max_{j \in [M]} \sum_{i=1}^{N} a_{ij} \bar{d}_i, \tag{4.20}$$

$$r_{\max} = \max_{k \in [K]} \sum_{i=1}^{N} p_{ik} \bar{d}_i + \Lambda q_{\max}, \tag{4.21}$$

$$G = \sqrt{M} \max \left\{ q_{\max}, \frac{I_{\max}}{T} \right\} \tag{4.22}$$

where $q_{\max}$ is the maximum resource consumption in each period for any resource $j \in [M]$; $r_{\max}$ is the maximum pseudo revenue that takes into account both the price of each product and the unit cost of each resource; and G is the maximum sub-gradient, i.e., Lipschitz parameter, of the dual objective function $g_t(\lambda)$. Recall from (4.4) that Λ is the bound of the dual values given by $\Lambda = (I_{\max}/I_{\min}) \cdot \sum_{j=1}^{M} p^j_{\max}$ where $I_{\min} = \min_{j \in [M]} I_j$, $I_{\max} = \max_{j \in [M]} I_j$, and $p^j_{\max} = \max_{i: a_{ij} \neq 0, k \in [K]} (p_{ik}/a_{ij})$.

Theorem 4.1. *The Bayesian regret of* FastTS *is bounded by*

$$\mathit{BayesRegret}(T) \ \leq 18 r_{\max}\sqrt{KT\log K} + \frac{3\sqrt{2}}{2}G\Lambda\sqrt{T} + q_{\max}\Lambda\left(\sqrt{T\log T}+1\right). \tag{4.23}$$

Proof outline. The regret analysis of algorithm FastTS is based on a primal-dual framework. Given dual multipliers λ_j for $j \in [M]$, each associated with a resource constraint, we can define the Lagrangian dual problem of $\mathsf{LP}(\theta)$ by the following LP, denoted by $\mathsf{LP}(\lambda, \theta)$.

$$\mathsf{LP}(\lambda,\theta): \ \max_x \sum_{k=1}^{K}\left(\sum_{i=1}^{N}\left(p_{ik} - \sum_{j=1}^{M}\lambda_j a_{ij}\right) d_{ik}(\theta)\right)x_k + \sum_{j=1}^{M}\lambda_j\frac{I_j}{T} \tag{4.24}$$

$$\text{s.t.} \ \sum_{k=1}^{K} x_k \leq 1, \tag{4.25}$$

$$x_k \geq 0, \ \forall k \in [K]. \tag{4.26}$$

Let the optimal value of $\mathsf{LP}(\lambda,\theta)$ be $\mathsf{OPT}(\lambda,\theta)$. By weak duality, we have $\mathsf{OPT}(\theta) \leq \mathsf{OPT}(\lambda(t),\theta)$. Denote the optimal solution of $\mathsf{LP}(\lambda(t),\theta)$ by $x^*(t)$, and the pricing decision of our algorithm in period t as $x(t)$, i.e., $x_k(t) = \mathbf{1}(k = k(t))$. We can decompose $\mathsf{OPT}(\lambda(t),\theta)$ by $\mathsf{OPT}(\lambda(t),\theta) = R(t,\theta)+\Delta_1(t,\theta)+\Delta_2(t,\theta)+\Delta_3(t,\theta)$, where we define

$$R(t,\theta) = \sum_{k=1}^{K}\left(\sum_{i=1}^{N} p_{ik} d_{ik}(\theta)\right)x_k\,, \tag{4.27}$$

$$\Delta_1(t,\theta) = \sum_{k=1}^{K}\left(\sum_{i=1}^{N}\left(p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right)d_{ik}(\theta)\right)\left(x_k^* - x_k(t)\right)\,, \tag{4.28}$$

$$\Delta_2(t,\theta) = \sum_{j=1}^{M}\left(\frac{I_j}{T} - \sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ij}D_i(t)\right)x_k\right)\lambda_j(t)\,, \tag{4.29}$$

$$\Delta_3(t,\theta) = \sum_{j=1}^{M}\left(\sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ij}\left(D_i(t) - d_{ik}(\theta)\right)\right)x_k\right)\lambda_j(t). \tag{4.30}$$

Suppose the algorithm ends in period τ. By the decomposition, we obtain

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}\left(\tau \geq t\right) \mathsf{OPT}(\theta)\right] \leq \mathbb{E}\left[\sum_{t=1}^{\tau} R(t,\theta) + \Delta_1(t,\theta) + \Delta_2(t,\theta) + \Delta_3(t,\theta)\right]. \quad (4.31)$$

By definition, $\sum_{t=1}^{\tau} R(t,\theta)$ denotes the total revenue achieved by the algorithm, $\sum_{t=1}^{\tau} \Delta_1(t,\theta)$ corresponds to the regret in the *primal* pricing decision space, $\sum_{t=1}^{\tau} \Delta_2(t,\theta)$ corresponds to the regret in the *dual* resource value space, and $\sum_{t=1}^{\tau} \Delta_3(t,\theta)$ corresponds to the stochastic error in demand, which connect the primal space and the dual space.

To show the regret bound in (4.23), we need to provide the upper bound for each Δ component respectively. Specifically. we will show

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \Delta_1(t,\theta)\right] \leq 18 r_{\max}\sqrt{KT\log K}, \quad (4.32)$$

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \Delta_2(t,\theta)\right] \leq \mathbb{E}\left[(\tau - T)\,\mathsf{OPT}(\theta)\right] + \frac{3\sqrt{2}}{2} G\Lambda\sqrt{T}, \quad (4.33)$$

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \Delta_3(t,\theta)\right] \leq q_{\max}\Lambda\left(\sqrt{T\log T} + 1\right). \quad (4.34)$$

Remark. The Bayesian regret bound of the LP-based Thompson sampling algorithm $\mathsf{TS\text{-}fixed}$ that is proposed in Ferreira et al. (2018) is

$$\text{BayesRegret}(T) \leq \left(18 p_{\max} + 38 \sum_{j=1}^{M}\sum_{i=1}^{N} p_{\max}^{j} a_{ij} \bar{d}_i\right)\sqrt{KT\log K}$$

where $\bar{d}_i$ the upper bound of $D_i(t)$, $p_{\max}^j := \max_{i:a_{ij}\neq 0, k\in[K]} \left(p_{ik}/a_{ij}\right)$ and $p_{\max} = \max_{k\in[K]} \sum_{i=1}^{N} p_{ik}\cdot \bar{d}_i$. See Theorem 1 in Ferreira et al. (2018). Compare our algorithm's Bayesian regret with the result in Ferreira et al. (2018). We observe that both algorithms obtain a Bayesian regret of order $O(\sqrt{TK\log K})$, and the constants are upper bounded by the same order of dependence in M and N. Moreover, both results are *prior-free* as they do not depend on the prior distribution of parameter θ, and the constants can be computed without knowing the demand distribution.

4.3.3 Numerical Experiments

In this section, we numerically compare our algorithm FastTS with other algorithms in the literature through several examples. When measuring the performance of an algorithm, we focus on the following two dimensions: (i) the algorithm's average revenue and (ii) the algorithm's average running time. These two dimensions respectively represent an algorithm's optimality and computational efficiency. For consistency, the examples we use are identical to the ones in the numerical section of Ferreira et al. (2018) and Besbes Zeevi (2012).

Single-Product Example

Consider a retailer selling a single product ($N = 1$) that consumes a single resource ($M = 1$) throughout a finite selling horizon of T periods. The set of feasible prices is $\{\$29.9, \$34.9, \$39.9\,\$44.9\}$, and the mean demand is $d(\$29.9) = 0.8, d(\$34.9) = 0.6, d(\$39.9) = 0.3, d(\$44.9) = 0.1$. As is common in revenue management literature, we show numerical results in an asymptotic regime when the inventory is scaled linearly with time. Given $I = \alpha T$, we consider two scenarios of initial inventory where $\alpha = 0.25$ and $\alpha = 0.5$.

We evaluate and compare the following dynamic pricing algorithms.

- FastTS: defined in Algorithm 4. We use the independent Beta prior as in Example 1.
- TS-Fixed: the algorithm proposed in Ferreira et al. (2018), which solves $\mathsf{LP}(t)$ with $c_j = I_j/T$. See Section 4.2.4. We use the independent Beta prior as in Example 1.
- TS-Updated: the algorithm proposed in Ferreira et al. (2018), which solves $\mathsf{LP}(t)$ with $c_j(t) = I_j(t-1)/(T-t+1)$. See Section 4.2.4. We use the independent Beta prior as in Example 1.
- BZ: the algorithm proposed in Besbes Zeevi (2012), which first explores all prices and then exploits the best pricing decisions by solving an LP once. In our implementation, we divide the exploration and exploitation phases at period

$\tau = T^{2/3}$ as suggested in their paper.

- PD-BwK: the algorithm proposed in Badanidiyuru et al. (2013) that uses the UCB algorithm to estimate demand $d(t)$ and uses a primal dual algorithm to solve LP(t). See the description in Section 4.2.5.
- TS: the original Thompson sampling algorithm presented in Thompson (1933), which does not consider inventory constraints.

We measure the optimality performance of an algorithm using the average percentage of "optimal revenue" achieved over 500 simulations. By "optimal revenue", we are referring to the optimal value of LP(θ), described in Section 4.3.2, and we know the optimal value is an upper bound on the optimal revenue where the retailer knows the true demand parameter θ prior to the selling season. In terms of computation performance, we use the average running time of an algorithm over the 500 simulations to describe the algorithm's computational efficiency. Figure 4-1 and Figure 4-2 show the performance results for the two scenarios $\alpha = 0.25$ and $\alpha = 0.5$, respectively.

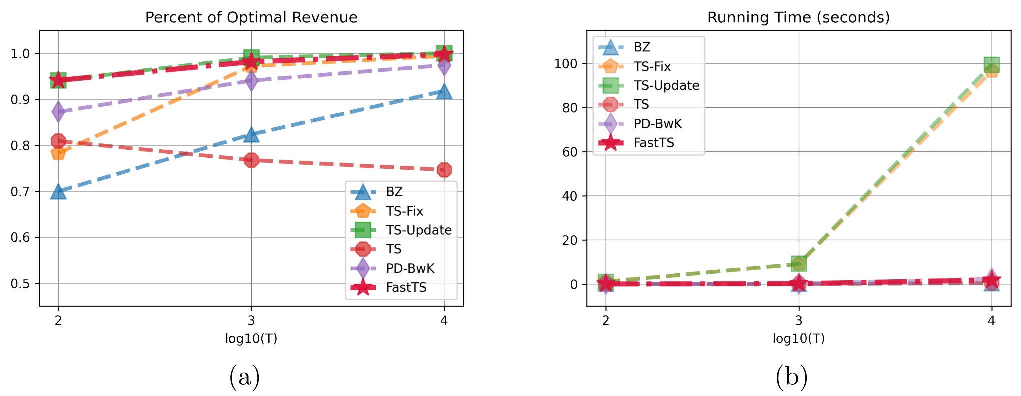


Figure 4-1: Performance comparison of different algorithms: single product example with $I = 0.25T$.

First, we notice that all the tested algorithms that takes into inventory constraints i.e., BZ, TS-Fixed, TS-Updated, PD-BwK, FastTS, converge to the optimal revenue as the length of the selling season increases, while algorithm TS does not. This is because the optimal strategies in both examples are a mixed strategy: when $I = 0.25T$, the optimal pricing policy is to offer \$39.90 to 3/4 of the customers, and offer \$44.90 to the remaining 1/4 of the customers; and when $I = 0.5T$, the optimal pricing

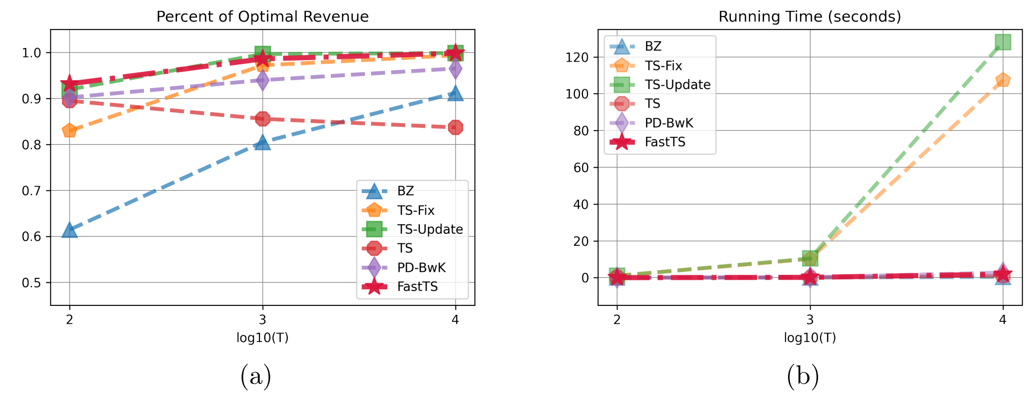


Figure 4-2: Performance comparison of different algorithms: single product example with $I = 0.5T$.

policy is to offer \$34.90 to 2/3 of the customers, and offer \$39.90 to the remaining 1/3 of the customers. In both cases, TS converges to the suboptimal price \$29.90 since this is the price that maximizes the expected revenue given unlimited inventory. This convergence result illustrates the necessity of incorporating inventory constraints when developing a dynamic pricing and learning algorithm.

Second, we notice that TS-Updated, TS-Fixed and our algorithm FastTS outperform other algorithms in terms of optimality. In particular, TS-Updated and FastTS show the best optimality performance. In addition, the optimality gap between any two different algorithms increases when (i) the length of the selling season is short and (ii) the ratio I/T is small. This shows that a good algorithm is able to quickly learn the true demand parameter and identify the optimal pricing strategy, which is very useful for low inventory settings.

Last, we notice from the running time comparison results that algorithms TS-Updated and TS-Fixed have a very long running time because both algorithms require to solve an estimated LP in the beginning of each selling period. For other algorithms, BZ only need to solve an LP once at the beginning of the exploitation phase, and algorithms FastTS and PD-BwK are *LP-free* since they are both based on a primal-dual framework. The running time of these three algorithms are much smaller than that of TS-Updated and TS-Fixed. Moreover, the performance gap in computational efficiency between the LP-heavy algorithms and LP-light algorithms becomes more significant as the length of the selling horizon increases.

If we evaluate an algorithm comprehensively from both dimensions, namely, optimality and computational efficiency, we observe that FastTS has the most outstanding performance. In terms of optimality, the performance of FastTS closely follows the algorithm with the highest average revenue, namely, TS-Updated. In terms of computational efficiency, the running time of FastTS is as low as the BZ and TS, which are the ones that have the lowest computational cost. In other words, among all the algorithms we test, no single algorithm can outperform FastTS in both dimensions at the same time.

Multi-Product Example

Consider a retailer selling two products ($N = 2$) that consume three resources ($M = 3$). The consumption of resources for each product, also known as the *bill-of-materials*, is as follows: one unit of product $i = 1$ consumes 1 unit of resource $j = 1$ and 3 units of resource $j = 2$; one unit of product $i = 2$ consumes 1 unit of resource $j = 1$, 1 unit of resource $j = 2$, and 5 units of resource $j = 3$. The set of feasible prices is $(p_1, p_2) \in \{(1, 1.5), (1, 2), (2, 3), (4, 4), (4, 6.5)\}$. We assume customers arrive according to a multivariate Poisson process. We consider the following types of demand functions.

1. Linear: $d(p_1, p_2) = (8 - 1.5p_1, 9 - 3p_2)$,
2. Exponential: $d(p_1, p_2) = (5e^{-0.5p_1}, 9e^{-p_2})$,
3. Logit: $d(p_1, p_2) = \left(\frac{10e^{-p_1}}{1+e^{-p_1}+e^{-p_2}}, \frac{10e^{-p_2}}{1+e^{-p_1}+e^{-p_2}}\right)$.

For each type of demand function, we consider two scenarios of initial inventory $I = \alpha T$ where $\alpha = (3, 5, 7)$ and $\alpha = (15, 12, 30)$, respectively.

We compare algorithms BZ, TS-Fixed, TS-Updated and our algorithm FastTS for this example. As mentioned in Ferreira et al. (2018), algorithm PD-BwK that is proposed in Badanidiyuru et al. (2013) does not apply to the setting in which customers arrive according to a Poisson process, so we do not include this algorithm in the comparison. For Thompson sampling based algorithms, we use the independent Gamma prior as described in Example 1.

As in the single-product example, we measure the optimality performance of an algorithm using the average percentage of“optimal revenue" achieved over 500 simulations, and measure the computational performance of an algorithm using the average running time of an algorithm over the 500 simulations to describe the algorithm's computational efficiency. In the following plots, Figure 4-3 and Figure 4-4 show the performance results for the linear demand function with $\alpha = 3$ and $\alpha = 15$, respectively. Figure 4-5 and Figure 4-6 show the performance results for the exponential demand function with $\alpha = 5$ and $\alpha = 12$. Figure 4-7 and Figure 4-8 show the performance results for the logit demand function with $\alpha = 7$ and $\alpha = 30$.

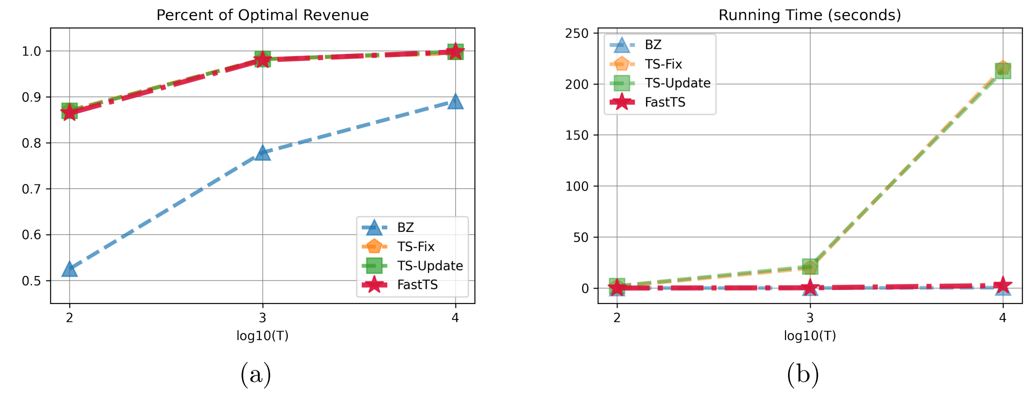


Figure 4-3: Performance comparison of different algorithms: multiple product linear demand with $I = 3T$.

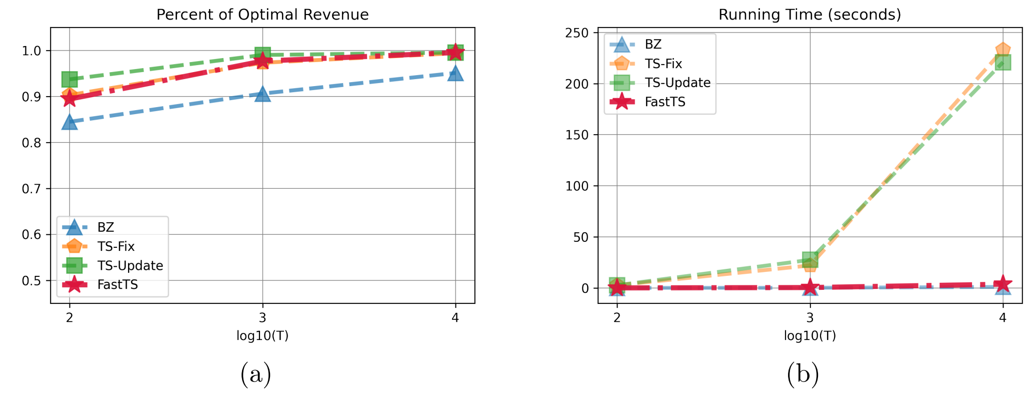


Figure 4-4: Performance comparison of different algorithms: multiple product linear demand with $I = 15T$.

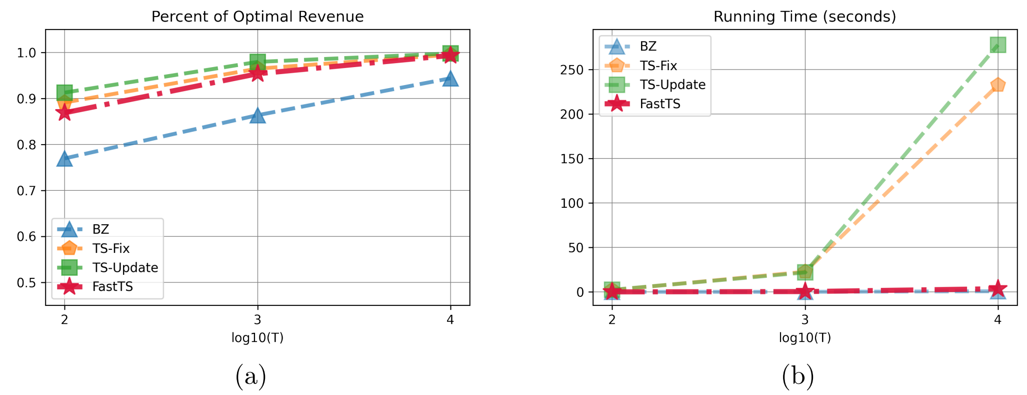


Figure 4-5: Performance comparison of different algorithms: multiple product exponential demand with $I = 5T$.

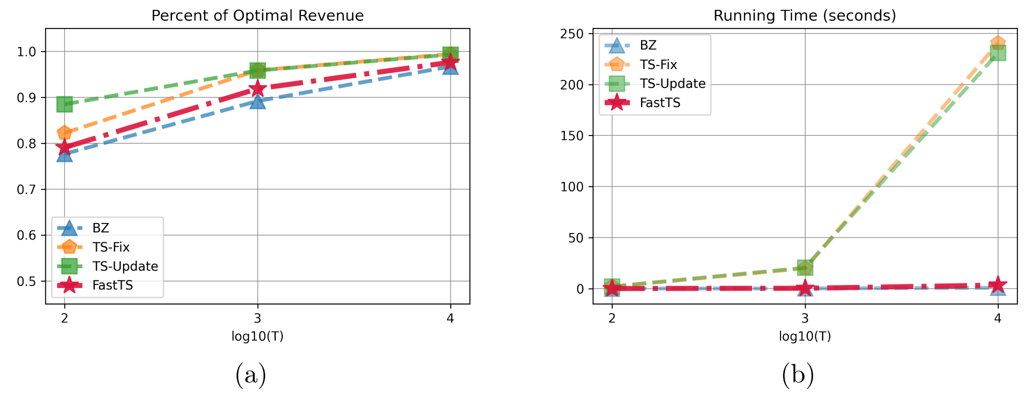


Figure 4-6: Performance comparison of different algorithms: multiple product exponential demand with $I = 12T$.

The experiments results for these multiple product examples under different demand functions are consistent with our observations in the single product example. In the optimality dimension, algorithm TS-Updated provides the best performance in most of the cases (except for the case of logit demand with $I = 30T$), followed by algorithm TS-Fixed and our algorithm FastTS. In the computational efficiency dimension, algorithm BZ provides the best performance, followed by algorithm FastTS. In a comprehensive evaluation of both dimensions, FastTS performs the best.

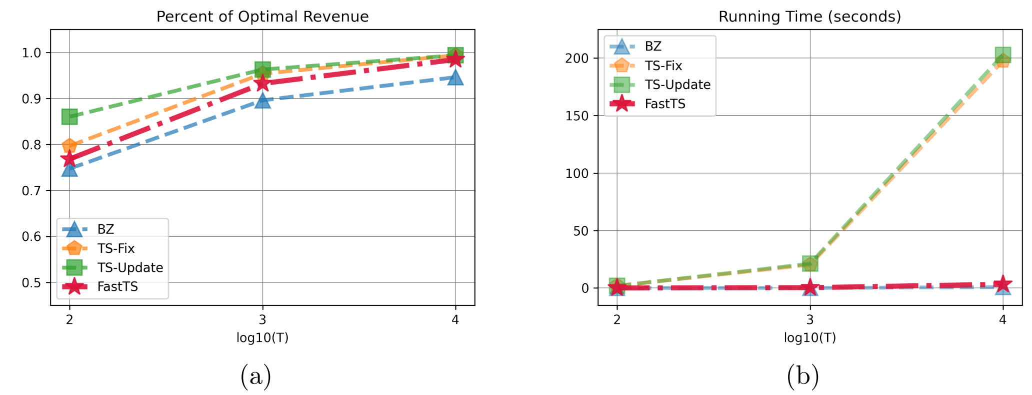


Figure 4-7: Performance comparison of different algorithms: multiple product logit demand with $I = 7T$.

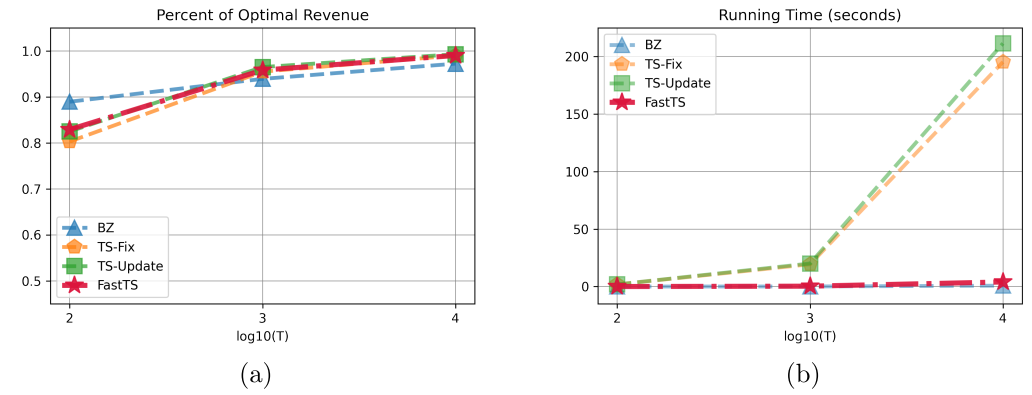


Figure 4-8: Performance comparison of different algorithms: multiple product logit demand with $I = 30T$.

4.4 Extensions and Further Applications

We discuss in this section extensions of our primal-dual algorithm and the corresponding analysis framework to a variety of settings in online network revenue management, and show the broad applicability of our approach in practice.

4.4.1 Contextual Model

We can extend our model and algorithm to the contextual setting where the retailer has access to some exogenous information, including customer attributes, product features, seasonality, etc, and can customize the pricing decisions based on such contextual information.

Model and Algorithm. Suppose that at the beginning of each time period $t \in [T]$, the retailer observes a context or feature $\xi(t)$. We assume $\xi(t)$ belongs to a discrete set $\mathcal{X}$, and is sampled i.i.d. from the set under a *known* probability distribution. After observing the context $\xi(t)$, the retailer selects a price vector $P(t) \in \{p_1, \ldots, p_K, p_\infty\}$, and then observes demand $D(t)$. Given any $\xi \in \mathcal{X}$, we assume that the demand under price vector p_k for any $k \in [K]$ is i.i.d. sampled from distribution with CDF $F(x_1, \ldots, x_N; p_k, \xi, \theta)$. The distribution is parametrized by an unknown vector $\theta \in \Theta$. We denote by $d_{ik}(\xi \mid \theta)$ the mean demand of product $i \in [N]$ under price vector p_k given context ξ and parameter θ. We also assume that the retailer knows a prior distribution of θ at the beginning of the selling season. The retail's objective is to maximize the expected revenue over the entire selling horizon.

We present our algorithm Fast-TS-Context for this model in Algorithm 6. In the initialization step, we define the domain of the dual variables λ, which is the same as shown in (4.4). When the selling season starts, in Step 1, we first sample demand parameter $\theta(t)$ according to its posterior probability distribution, and calculate the mean demand estimation $d_{ik}(\xi(t))$ given observation $\xi(t)$. In Step 2, we optimize the price by selecting the price vector that maximizes the pseudo revenue given $d_{ik}(\xi(t))$. Here, the definition of pseudo revenue follows that in Algorithm 4. After observing customer's demand $D(t)$, we update the posterior distribution of θ in Step 3, and update the dual variable $\lambda(t)$ by the online gradient descent algorithm described in Algorithm 5 in Step 4. The definition of the dual function also follows that of Algorithm 4.

Note that Fast-TS-Context follows the same framework as FastTS. The only difference lies in the *primal space*, where we make several simple modifications in the Thompson sampling steps and the price optimization step to include the contextual information $\xi(t)$. The update of the dual variables is not changed since the *dual space* of the problem stays the same. In fact, the problem defined in Section 4.2.1 can be considered as a special case of this contextual model where we have $|\mathcal{X}| = 1$.

Algorithm 6 Fast Thompson Sampling for Contextual Pricing (Fast-TS-Context)

```
0. Initialization. Set $\lambda_j(0) = 0$ and $I_j(0) = I_j$ for all $j \in [M]$. For each time period $t = 1, \dots, T$:
1. Sample demand. Sample $\theta(t)$ from $\Theta$ according to the posterior distribution of $\theta(t)$ given history $\mathcal{H}_{t-1}$.
2. Optimize pricing decisions. Observe context $\xi(t)$. Let $d_{ik}(\xi(t) \mid \theta(t))$ be the mean demand given $\theta(t)$. Offer the optimal price vector with index $k(t)$ such that
       $k(t) = \arg\max_{k \in [K+1]} \sum_{i=1}^{N} \left( p_{ik} - \sum_{j=1}^{M} \lambda_j(t) a_{ij} \right) d_{ik}(\xi(t) \mid \theta(t)).$   (4.35)
3. Update estimate of parameter $\theta$. Observe demand $D(t)$. Update the history $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{\xi(t), P(t), D(t)\}$ and the posterior distribution of $\theta$ given $\mathcal{H}_t$.
4. Update parameter $\lambda$. Define function
       $g_t(\lambda) = \sum_{j=1}^{M} \lambda_j \cdot (I_j/T - \sum_{i=1}^{N} a_{ij} D(t)).$   (4.36)
   Update $\lambda(t+1)$ by Online Gradient Descent with step size $\eta_t = C/\sqrt{t}$.
5. Stopping condition. Update the inventory of each resource $j \in [M]$. If there exists $j$ such that inventory $I_j(t) \le 0$, then Exit.
```

Performance Analysis. The Bayesian regret bound of Fast-TS-Context is provided in Theorem. 4.2. Comparing this results with that in Theorem. 4.1, we observe that the regret from the dual update procedure and the stochastic errors in $D(t)$ stay the same. The only difference of analysis is in the primal space, where we need to take into account context $\xi(t)$. In addition, the result in Theorem. 4.2 also shows the same order of regret as in Theorem 4 of Ferreira et al. (2018) where the authors consider an extension of algorithm TS-Fixed to the contextual setting.

Theorem 4.2. *The Bayesian regret of Fast-TS-Context is bounded by*

$$\mathit{BayesRegret}(T) \le 18 r_{\max}\sqrt{|\mathcal{X}| K T \log K} + \frac{3}{2} G \Lambda \sqrt{T} + q_{\max}\Lambda \left(\sqrt{T \log T} + 1 \right). \quad (4.37)$$

We notice that the regret bound in Theorem. 4.2 depends on the size of the feature space by $O(\sqrt{|\mathcal{X}|})$, and is meaningful when $|\mathcal{X}|$ is small compared with T. The result is also compatible with the regret bound shown in Agrawal et al. (2016b) for the contextual bandit problem with resource constraints. Specifically, given the set of admissible policies that maps $\mathcal{X}$ to $\{p_1, \dots, p_k\}$, the regret is upper bounded by $O(\sqrt{KT\log(T|\Pi|)})$. Take $|\Pi| = K^{|\mathcal{X}|}$, and we recover the regret bound in our theorem (up to log factors).

The proof of Theorem. 4.2 shares a similar analysis structure to Theorem. 4.1, as discussed in Section 4.3.2. The benchmark is now

$$\mathsf{LP}(\theta): \max_x \mathbb{E}_\xi \left[\sum_{k=1}^K \left(\sum_{i=1}^N p_{ik} d_{ik}(\xi|\theta) \right) x_{\xi,k} \right] \tag{4.38}$$

$$s.t.\ \mathbb{E}_\xi \left[\sum_{k=1}^K \left(\sum_{i=1}^N a_{ij} d_{ik}(\xi|\theta) \right) x_{\xi,k} \right] \le \frac{I_j}{T},\ \forall j \in [M] \tag{4.39}$$

$$\sum_{k=1}^K x_{\xi,k} \le 1,\ \forall \xi \in \mathcal{X} \tag{4.40}$$

$$x_{\xi,k} \ge 0,\ \forall \xi \in \mathcal{X},\ \forall k \in [K]. \tag{4.41}$$

where we use $\mathbb{E}_\xi[\cdot]$ to denote the expectation over context $\xi(t)$ given its known probability distribution. This expectation appears both in the objective and in the constraints.

4.4.2 Linear Contextual Model

In this section, we extend our model and algorithm to the linear contextual setting where we assume demand is a linear function of context. In many practical applications, context $\xi(t)$ is defined on a high-dimensional space $\mathbb{R}^d$ rather than a discrete set, and thus the number of contexts $|\mathcal{X}|$ can hardly be bounded, rendering the results in the previous section meaningless. In this case, we can adopt a linear model to parameterize a customer's demand under different prices.

Model and Algorithm. Let $y(\xi, k)$ denote the function that maps the contextual information associated with context $\xi \in \mathcal{X}$ and price vector p_k for $k \in [K]$ to a vector in $\mathbb{R}^d$. In each time period $t \in [T]$, given observed context $\xi(t)$ and selected price vector $k(t)$, define $y(t) = y(\xi(t), k(t))$. We assume that $D(t) = W^\top y(t) + \varepsilon(t)$ where W is a parameter matrix in $\mathbb{R}^{d \times N}$, and $\varepsilon(t)$ is a demand noise vector in $\mathbb{R}^N$. We also assume $\varepsilon(t)$ is sampled independently from a known multivariate normal distribution with zero mean. The demand parameter $\theta = W$ is unknown, and the retailer knows a prior distribution of θ over Θ.

We present our algorithm Fast-TS-LinContext for this model in Algorithm 7. The algorithm shares the same framework as Algorithm 4 and Algorithm 6. The only difference is that in step 2, we calculate the mean demand $d_{ik}(\xi(t) \mid \theta(t))$ under each price vector p_k using linear function $\theta^\top y(t)$ given context $y(t) = (\xi(t), k)$.

Algorithm 7 Fast Thompson Sampling for Contextual Pricing (Fast-TS-Lin-Context)

0. Initialization. Set $\lambda_j(0) = 0$ and $I_j(0) = I_j$ for all $j \in [M]$. For each time period $t = 1, \dots, T$:

1. Sample demand. Sample $\theta(t)$ from Θ according to the posterior distribution of $\theta(t)$ given history $\mathcal{H}_{t-1}$.

2. Optimize pricing decisions. Observe context $\xi(t)$. Let $d_{ik}(\xi(t) \mid \theta(t))$ be the mean demand given $\theta(t)$. Offer the optimal price vector with index $k(t)$ such that

$$k(t) = \arg\max_{k \in [K+1]} \sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j(t) a_{ij} \right) d_{ik}(\xi(t) \mid \theta(t)). \tag{4.42}$$

3. Update estimate of parameter θ. Observe demand $D(t)$. Update the history $\mathcal{H}_t = \mathcal{H}_{t-1} \cup \{\xi(t), P(t), D(t)\}$ and the posterior distribution of θ given $\mathcal{H}_t$.

4. Update parameter λ. Define function

$$g_t(\lambda) = \sum_{j=1}^{M} \lambda_j \cdot (I_j/T - \sum_{i=1}^{N} a_{ij} D(t)). \tag{4.43}$$

Update $\lambda(t+1)$ by Online Gradient Descent with step size $\eta_t = C/\sqrt{t}$.

5. Stopping condition. Update the inventory of each resource $j \in [M]$. If there exists j such that inventory $I_j(t) \le 0$, then Exit.

Note that in the model, we assume a discrete price set, and thus we can optimize price decisions by sorting different choices of prices without solving any LPs. If we assume the retailer chooses prices from a continuous price set, we may need to run the price optimization via a linear program. For example, Ferreira et al. (2018) consider a context-independent setting where price $P(t) = [P_i(t)]_{i \in [N]}$ is from a bounded polyhedral set $\mathcal{P}$, and demand is given by linear function $D(t) = \alpha + BP(t) + \varepsilon(t)$ with $(\alpha, B) \subset \mathbb{R}^{N \times (N+1)}$. In this setting, we have in step 2 the following optimization problem in period t:

$$P(t) = \arg\max_{p \in \mathcal{P}} \sum_{i=1}^{N} \left(p - \sum_{j=1}^{M} \lambda_j(t) a_{ij} \right) (\alpha(t) + B(t)p), \tag{4.44}$$

where $\alpha(t)$ and $B(t)$ are sampled from their posterior distributions. Since this optimization problem does not involve any inventory constraints, we can expect that our primal-dual algorithm framework still has the advantage in computational efficiency if the feasible price set $\mathcal{P}$ is not too complicated.

Performance Analysis. The Bayesian regret bound of Fast-TS-Lin-Context is provided in Theorem. 4.3. In comparison with the result in Theorem. 4.1, we observe that the only difference is from the regret of the primal decisions.

Theorem 4.3. *The Bayesian regret of Fast-TS-Lin-Context is bounded by*

$$BayesRegret(T) \le r_{\max} O\left(dN \log T \sqrt{T}\right) + \frac{3}{2} G \Lambda \sqrt{T} + q_{\max} \Lambda \left(\sqrt{T \log T} + 1\right). \tag{4.45}$$

We highlight that the regret bound of algorithm Fast-TS-Lin-Context does not depend on the number of price vectors and the number of contexts, but on the dimension of the demand function class, which is amount to the number of unknown parameters in our case. Specifically, we know $\theta = W \subset \mathbb{R}^{d \times N}$, and this leads to the regret bound $O\left(dN \log T \sqrt{T}\right)$ in (4.45).

The proof of Theorem. 4.3 follows the same analysis framework as Theorem. 4.2. In the analysis, we build a connection between the dynamic pricing problem and the multi-armed bandit problem where the reward of choosing price vector k is given by the pseudo revenue $\sum_{i=1}^{N}\left(p_{ik}-\sum_{j=1}^{M}\lambda_j(t)a_{ij}\right)d_{ik}(\xi(t)\mid\theta(t))$, with $d(\xi(t)\mid\theta(t))$ being a linear function in context $y(t)=(\xi(t),k)$. We build connection between our problem with the linear bandit problems in Rusmevichientong Tsitsiklis (2010), Agrawal Devanur (2016) and Russo Van Roy (2014) to analyze the regret bound of our algorithm.

Chapter 5

Concluding Remarks

In this chapter, we summarize the previous works and discuss future research directions based on the models and algorithms we study in each chapter.

5.1 Summary and Future Research Directions

Online matching with Bayesian rewards

We study in Chapter 2 an online matching problem where a central platform needs to match finite resources to users that arrive sequentially over time. The reward of each matching option depends both on the the type of resource and the time period the user arrives. The matching rewards are assumed to be unknown, but the prior and posterior updating rules are known a priori. The goal is to maximize the total reward from all matching options without violating the resource capacity constraints.

The matching problem features a Bayesian learning environment and a non-stationary reward distribution over time. In addition, the budget constraints are described both by the number of available impressions of each resource and by the number of users in each time period. The main contribution of the paper is that we propose an algorithm for the matching problem that achieves a constant fraction of the optimal reward. In particular, our algorithm is based on smartly assembling the single-arm policies that are obtained from the solutions to a linear program. In the technical analysis, we show that our algorithm improves the previous results in Guha Munagala (2013), and we also provide geometric intuitions in proving the algorithm's performance guarantee.

Future research directions. The algorithm framework we developed in the chapter can be further extended. First, in the work, we consider deterministic arrivals, and we believe the techniques can be extended to the setting of stochastic arrivals. In fact, there exist tools in the online matching literature that can help us handle the stochastic setting. Second, in the work, we aim to provide an approximation ratio guarantee for the non-stationary learning problem, while existing literature mostly focuses on using regret analysis. Each of the two performance metrics has its advantages and disadvantages, and we believe there are connections we can explore between these two metrics. This topic also relates to the similarities and differences between Markovian bandits and stochastic bandits. Moreover, we believe a tightness analysis on the current performance ratio guarantee can make our results more convincing. Last, in the matching problem, we focus on the efficiency of the central platform and the goal to maximize the total rewards. In practice, the fairness between different resource channels may also be a major concern to the central platform. In this case, it is worth exploring the mechanism design of the allocation platform and the operational restrictions that can help improve the efficiency of all resource channels. The impact of such fairness constraints in the online matching problem is an interesting future research direction.

Online learning and optimization for add-on discounts

We study in Chapter 3 a revenue management problem with add-on discounts, which is motivated by the unique structure between core products (video game consoles) and supportive products (video games). We note that although the add-on discount strategy has been used in the industry, it has not been formally studied in the literature, and our work fills this gap between theory and practice. In particular, we develop an optimization formulation of the revenue management problem, and provide an FPTAS algorithm that can approximately solve the optimization problem to any desired accuracy. Moreover, we study the problem in the online setting where the demand functions of different products are unknown. We propose a UCB-based algorithm to solve the online problem, and show that the algorithm can obtain a tight regret bound.

The results of this study provides useful managerial insights and strategical guidance for retailers. In principle, the add-on discount strategy offers more flexibility for product promotions, and retailers can increase their revenue (and profit) by adopting this strategy so as to incentivize customers to purchase more items. However, in practice, the lack of past experience and the uncertainty of customer's demand could hold retailers back from implementing the strategy. In our numerical experiments, which are based on the real-world data we collect from Tmall.com, we show that the retailer can expect a revenue (profit) increase of 5% to 20% by using add-on discounts. More importantly, in the more practical setting where the retailer has no prior knowledge of the demand information, we show that the retailer can obtain a long-term increase in revenue (profit) by using the add-on discount strategy while learning the demand parameters on the fly. These numerical results demonstrate the efficacy of using data-driven approaches in revenue management.

Future research directions. We point out several interesting future research directions based on our work. First, our model motivates a more general add-on setting where discounts are offered *two-way.* In this work, we categorize the products into core products and supportive products, and assume that the retailer can only offer add-on discounts on supportive products. In the more general setting, given two or more selected sets of products, we may assume that the retailer can offer add-on discounts to any set of products. Furthermore, in general product bundles, how to model the relationship between different types of products is more challenging. In the example of games and game consoles, we have various selections on the supportive products that are all compatible with the core product. This might not be the case when we consider other examples like printers and inks, razors and replacement blades, etc.

Second, building on the results of this paper, it worth exploring another innovative revenue management strategy called *share-for-discounts*. In share-for-discounts, customers can collect bonus points by sharing the information of certain products with their friends. Once the bonus points reach some threshold, a customer can get discounts on the shared products as rewards. By using this strategy, retailers can reach more potential customers through a customer's personal social network. Therefore, how to design a good data-driven policy for the share-for-discounts strategy would be another interesting research direction.

Fast Thompson sampling for online network revenue management

We study in Chapter 4 a canonical price-based network revenue management problem where a retailer aims to maximize revenue from multiple products with limited inventory over a finite selling season. We assume the demand of different products, as functions of prices, contains unknown parameters, and the retailer must learn them from the sales data. The main contribution of the work is that we propose an primal-dual algorithm that combines the Thompson sampling technique and the online gradient descent method. We not only provide Bayesian regret bound of the algorithm, but also numerically show the algorithm's computational efficiency.

The primal-dual framework we develop in the work provides a convenient tool for both designing and analyzing algorithms. In particular, in the primal decision space, we aim to learn the demand parameter by formulating the problem as a multi-armed bandit, and in the dual decision space, we aim to learn the unit value of resource by formulating the problem as an online convex optimization problem. Since the framework does not require solving any LPs, our algorithm obtains the advantage in computational efficiency in comparison to other LP-based algorithms in the literature. In the work, we also discuss extensions of the algorithm framework to various contextual pricing settings.

Future research directions. We note that in our algorithm's primal-dual framework, we can easily replace the learning algorithms for both the the primal and the dual problems with other alternatives. For example, we can replace the Thompson sampling algorithm with UCB-type learning algorithms, and replace the online gradient descent algorithm with follow-the-leader (FTL) algorithm or follow-the-regularized-leader (FTRL) algorithm. It would be an interesting exercise to analyze the regret bounds of these algorithm variants, and show their numerical performances. In addition, in our algorithm's primal-dual framework, we consider the dual online optimization problem with fixed resource inventory parameters. It is worth exploring the dual problem with updated inventory values, and analyzing the performance of the corresponding algorithm. Last, we highlight that the application of our primal-dual algorithm framework is not limited to dynamic pricing, and it would be an interesting research direction to explore extensions of our framework to assortment optimization, online resource allocation, and all sorts of multi-armed bandit problems with resource constraints.

Appendix A

Technical Results in Chapter 2

A.1 Proof of Theorem 2.1

Proof. Consider an online algorithm that decides the action of pulling each arm based on the joint state of the $N \times T$ arms. Let $u_{i,t} \in S_{i,t}$ denote the state of arm (i,t), and $u = \prod_{i,t} u_{i,t} \in \prod_{i,t} S_{i,t}$ the joint state of the $N \times T$ arms. Let $X_{i,t}(u_{i,t}) \in \{0,1\}$ denote whether or not arm (i,t) ever enters state $u_{i,t}$ and is pulled while the arm is in state $u_{i,t}$, $Z_{i,t}(u_{i,t}) \in \{0,1\}$ the binary result of pulling arm (i,t) while the arm is in state $u_{i,t}$, and $Y_{i,t}(u_{i,t}) \in \{0,1\}$ whether or not arm (i,t) ever enters state $u_{i,t}$. The online algorithm can be written as the mapping between the joint state of $N \times T$ arms and the action $\prod_{i,t} X_{i,t}(u_{i,t})$, namely, $\prod_{i,t} u_{i,t} \to \prod_{i,t} X_{i,t}(u_{i,t})$, for all states $u_{i,t} \in S'_{i,t}$ with $\sum_{i,t} X_{i,t}(u_{i,t}) \leq 1$.

By definition, a feasible online algorithm satisfies the constraints:

$$\sum_{i\in[N]} \sum_{u_{i,t}\in S'_{i,t}} X_{i,t}(u_{i,t}) \leq D_t,$$

$$\sum_{t\in[T]} \sum_{u_{i,t}\in S'_{i,t}} X_{i,t}(u_{i,t}) \leq B_i,$$

$$X_{i,t}(u_{i,t}) \leq Y_{i,t}(u_{i,t}), \quad \forall u_{i,t} \in S'_{i,t}.$$

In addition, for each arm (i,t) with $i \in [N]$ and $t \in [T]$, the transition of the arm's associated MDP follows the constraints:

- for root state $\rho_{i,t} = (0,0)$,
$$Y_{i,t}(\rho_{i,t}) = 1,$$
- for state $u_{i,t} = (a,b) \in S''_{i,t}$ with $a > 0, b = 0$, and $v_{i,t} = (a-1,b)$,
$$Y_{i,t}(u_{i,t}) = X_{i,t}(v_{i,t}) Z_{i,t}(v_{i,t}),$$
- for state $u_{i,t} = (a,b) \in S''_{it}$ with $a = 0, b > 0$, and $w_{i,t} = (a, b-1)$,
$$Y_{i,t}(u_{it}) = X_{i,t}(w_{i,t})(1 - Z_{i,t}(w_{i,t})),$$
- for state $u_{i,t} = (a,b) \in S''_{i,t}$ with $a > 0, b > 0$, and $v_{i,t} = (a-1,b)$, $w_{i,t} = (a,b-1)$,
$$Y_{i,t}(u_{i,t}) = X_{i,t}(v_{i,t}) Z_{i,t}(v_{i,t}) + X_{i,t}(w_{i,t})(1 - Z_{i,t}(w_{i,t})).$$

The accumulative reward of the online algorithm is given by

$$\sum_{i \in [N]} \sum_{t \in [T]} \sum_{u_{i,t} \in S'_{i,t}} X_{i,t}(u_{i,t}) Z_{i,t}(u_{i,t}).$$

Note that $Z_{i,t}(u_{i,t}) \in \{0,1\}$ is a random variable. Moreover, due to the fact that $X_{i,t}(u_{i,t})$ and $Z_{i,t}(u_{i,t})$ are independent under the condition that arm (i,t) is in state $u_{i,t} \in S'_{i,t}$, we have

$$\begin{aligned}
&\mathbb{E}[X_{i,t}(u_{i,t}) \cdot Z_{i,t}(u_{i,t}) \mid Y_{i,t}(u_{i,t}) = 1] \\
=&\mathbb{E}[X_{i,t}(u_{i,t}) \mid Y_{i,t}(u_{i,t}) = 1] \cdot \mathbb{E}[Z_{i,t}(u_{i,t}) \mid Y_{i,t}(u_{i,t}) = 1] \\
=&\mathbb{E}[X_{i,t}(u_{i,t}) \mid Y_{i,t}(u_{i,t}) = 1] \cdot p^{(u)}_{i,t},
\end{aligned}$$

where $p^{(u)}_{i,t}$ follows the definition in Section 2.2.3, namely, $p^{(u)}_{i,t} := \int_0^1 p \cdot \theta^{(u)}_{i,t}(p)\ dp$. Recall that $\theta^{(u)}_{i,t}(p)$ is the density function of the posterior distribution of $p_{i,t}$ in state

$u \in S_{i,t}$. The expected reward of the online algorithm is

$$\sum_{i\in[N]}\sum_{t\in[T]}\sum_{u_{i,t}\in S'_{i,t}} p_{i,t}^{(u)} \cdot \mathbb{E}[\mathbb{E}[X_{i,t}(u_{i,t}) \mid Y_{it}(u_{i,t}) = 1]].$$

Now we define $x_{i,t}^{(u)}$ and $y_{i,t}^{(u)}$ as follows.

$$x_{i,t}^{(u)} := \mathbb{E}[\mathbb{E}[X_{i,t}(u_{i,t}) \mid Y_{i,t}(u_{i,t}) = 1]],\ y_{i,t}^{(u)} := \mathbb{E}[Y_{i,t}(u_{i,t})].$$

It can be easily verified that $x_{i,t}^{(u)}$ and $y_{i,t}^{(u)}$ are feasible solutions to LP. In addition, we observe that the objective function of LP is equal to

$$\sum_{i\in[N]}\sum_{t\in[T]}\sum_{u_{i,t}\in S'_{i,t}} p_{i,t}^{(u)} x_{i,t}^{(u)} = \sum_{i\in[N]}\sum_{t\in[T]}\sum_{u_{i,t}\in S'_{i,t}} p_{i,t}^{(u)} \mathbb{E}[\mathbb{E}[X_{i,t}(u_{it}) \mid Y_{i,t}(u_{it}) = 1]],$$

which is the exactly the expected reward of the online algorithm. Hence we know that the expected reward of any online algorithm, and therefore, the optimal online algorithm, is upper-bounded by the optimal value of LP.

□

A.2 Proof of Theorem 2.2

A similar theorem and its proof are provided in Guha Munagala (2013) Theorem 2 (The Truncation Theorem). Since we have a different setting in this paper, we restate the theorem and provide the proof as follows. The proof is built on the *Martingale Property* (see Lemma A.1), which implies the equivalence between the following two ways of accounting the expected reward and expected cost of each arm's single-arm policy. (For simplicity, we remove the subscript m.)

(i) Let S denote the underlying state space of the arm's MDP. Given single-arm policy $\mathcal{P}(\mathbf{x}, \mathbf{y}, K)$, we have

$$\mathcal{R}(\mathcal{P}(\mathbf{x}, \mathbf{y}, K)) := \sum_{u:(a,b)\in S} p^{(u)} x^{(u)} \mathbf{1}(a + b < K)$$

$$\mathcal{K}(\mathcal{P}(\mathbf{x}, \mathbf{y}, K)) := \sum_{u:(a,b)\in S} x^{(u)} \mathbf{1}(a + b < K)$$

(ii) Let q denote the "path", i.e., state evolution in state space S that corresponds to running single-arm policy $\mathcal{P}(\cdot, K)$ till stop. Let $\ell(q)$ denote the length of path q, i.e., the associated number of pulls. We know $\ell(q) \le K$ for any q. Let $g(p, q)$ denote the probability of seeing q when the true reward p takes value p. Let $\mathcal{R}(\mathcal{P}(\cdot), p)$ denote the expected reward and $\mathcal{K}(\mathcal{P}(\cdot, K), p)$ the expected cost of running single-arm policy when the true reward takes value p. Then we have

$$\mathcal{R}(\mathcal{P}(\cdot, K), p) = \sum_q p \cdot \ell(q) \cdot g(p, q) \ \text{ and } \ \mathcal{K}(\mathcal{P}(\cdot, K), p) = \sum_q \ell(q) \cdot g(p, q).$$

In addition, we obtain

$$\mathcal{R}(\mathcal{P}(\cdot, K)) = \mathbb{E}_p \left[\mathcal{R}(\mathcal{P}_{i,t}(\cdot, K), p)\right] \ \text{ and } \ \mathcal{K}(\mathcal{P}(\cdot, K)) = \mathbb{E}_p \left[\mathcal{R}(\mathcal{P}_{i,t}(\cdot, K), p)\right],$$

where p follows the prior probability distribution $\theta^{\rho}(\cdot)$.

Now consider single-arm policy $\mathcal{P}(\mathbf{x}, \mathbf{y}, \beta K)$. For each path q that $\mathcal{P}(\mathbf{x}, \mathbf{y}, K)$ encounters, policy $\mathcal{P}(\mathbf{x}, \mathbf{y}, \beta K)$ encounters the path with the very same probability, except that the run of $\mathcal{P}(\mathbf{x}, \mathbf{y}, \beta K)$ stops after *at least* β fraction of the total number of pulls. This means that for each path q, we have a truncated path with length $\ell'(q) \ge \beta \ell(q)$. Specifically, we have

- $\ell'(q) = \ell(q)$, if $\ell(q) \le \beta K$;
- $\beta \ell(q) \le \ell'(q) < \ell(q)$, if $\beta K < \ell(q) \le K$.

Correspondingly, we have

$$\mathcal{R}(\mathcal{P}(\cdot, \beta K), p) = \sum_q p \cdot \ell'(q) \cdot g(p, q) \ge \sum_q p \cdot \beta \ell(q) \cdot g(p, q),$$

$$\mathcal{K}(\mathcal{P}(\cdot, \beta K), p) = \sum_q \ell'(q) \cdot g(p, q) \le \sum_q \ell(q) \cdot g(p, q).$$

By linearity of expectation, after taking expectation over p on both sides, we have

$$\mathcal{R}(\mathcal{P}(\cdot, \beta K)) \geq \beta \cdot \mathcal{R}(\mathcal{P}(\cdot, K)),$$

$$\mathcal{K}(\mathcal{P}(\cdot, \beta K)) \leq \mathcal{K}(\mathcal{P}(\cdot, K)).$$

A.3 Proof of Theorem 2.3

By Proposition 2.1, we have for algorithm Prophet that

$$\mathsf{ALG} \geq \sum_{t \in [T]} \left(1 - \frac{1}{B} \sum_{s<t} \widetilde{\mathcal{K}}_s\right) \widetilde{\mathcal{R}}_t. \tag{A.1}$$

Following the discussion in Section 2.3.2, we know the right-hand side of (A.1) is lower-bounded by $(e_1 + e_2) \cdot \sum_t \mathcal{R}_t$. By Lemma 2.2, we have $(e_1 + e_2) \geq \sqrt{2} - 1$. Therefore, we obtain

$$\sum_{t \in [T]} \left(1 - \frac{1}{B} \sum_{s<t} \widetilde{\mathcal{K}}_s\right) \widetilde{\mathcal{R}}_t \geq \left(\sqrt{2} - 1\right) \sum_{t \in [T]} \mathcal{R}_t.$$

By Theorem 2.1, we know $\sum_t \mathcal{R}_t \geq \mathsf{OPT}$. Therefore, we obtain $\mathsf{ALG} \geq \left(\sqrt{2} - 1\right) \mathsf{OPT}$.

A.4 Proof of Theorem 2.4

Proof. Given algorithm TS-Prophet, let $n_{i,t}$ denote the number of times arm (i, t) is pulled. Recall that $B_i(t)$ is the remaining budget of resource i at the beginning of period t. By definition, we have

$$B_i(t) = B_i - \sum_{s<t} n_{i,s} \text{ for } t \in [T].$$

Given the set of active arms $(\mathcal{Q}_t, t)$ in time period t and $Q_t = |\mathcal{Q}_t|$, let $I_t(1), \ldots, I_t(Q_t)$ denote the order of running these Q_t prophet policies. Let $D_t(i)$ denote the remaining number of users ("budget") in time period t when arm (i,t) is pulled. By definition, we have for the k-th arm $(I_t(k),t)$ that

$$D_t(I_t(k)) = D_t - \sum_{j<k} n_{I_t(j),t} \text{ for } k \in [M_t].$$

Recall that the budget parameter of prophet policy $\widetilde{\mathcal{P}}_{i,t}(K_{i,t})$ at the beginning of time period t is given by $K_{i,t} = \min\{B_i(t), D_t\}$. Given arm $(i,t) \in \mathcal{Q}_t$, let $I_t^{-1}(i)$ denote the index of arm (i,t) in order $I_t(1),\ldots,I_t(Q_t)$. Then the budget parameter of prophet policy $\widetilde{\mathcal{P}}_{i,t}(K_{i,t})$ when arm (i,t) is pulled is given by $K'_{i,t} = \min\{B_i(t), D_t(i)\}$, where $D_t(i) := D_t - \sum_{j<I^{-1}(i)} n_{I_t(j),t}$.

Recall that $\widetilde{\mathcal{K}}'_{i,t} = \mathcal{K}(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\}))$.

Let $\widetilde{\mathcal{K}}_{i,t} := \mathcal{K}(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i, D_t\}))$. For arm (i,t), we have

$$\mathbb{E}\left[n_{i,t}\right] = \mathbb{E}\left[\mathbb{E}\left[n_{i,t} \mid K'_{i,t}\right]\right] \le \mathbb{E}\left[\mathcal{K}'_{i,t}\right] \le \mathbb{E}\left[\widetilde{\mathcal{K}}_{i,t}\right] = \widetilde{\mathcal{K}}_{i,t}, \tag{A.2}$$

where both inequalities follow from Theorem 2.2 (ii).

Recall that $\widetilde{\mathcal{R}}'_{i,t} = \mathcal{R}(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i(t), D_t\}))$, and $\mathcal{R}_{i,t} = \mathcal{R}(\mathcal{P}_{i,t}(\min\{B_i, D_t\}))$.

Let $\widetilde{\mathcal{R}}_{i,t} := \mathcal{R}(\widetilde{\mathcal{P}}_{i,t}(\min\{B_i, D_t\}))$. Let ALG denote the expected reward of algorithm $\mathsf{TS\text{-}Prophet}$.

We have

$$\begin{aligned}
\mathsf{ALG} &= \mathbb{E}\left[\sum_{t\in[T]}\sum_{i\in\mathcal{Q}_t} \mathcal{R}(\widetilde{\mathcal{P}}_{i,t}(K'_{i,t}))\right] \\
&\ge \mathbb{E}\left[\sum_{t\in[T]}\frac{1}{2}\sum_{i\in\mathcal{Q}_t} \widetilde{\mathcal{R}}'_{i,t}\right] \quad \text{by (2.13)} \\
&= \frac{1}{2}\mathbb{E}\left[\sum_{i\in[N]}\sum_{t:i\in\mathcal{Q}_t} \widetilde{\mathcal{R}}'_{i,t}\right] \\
&\ge \frac{1}{2}\mathbb{E}\left[\sum_{i\in[N]}\sum_{t:i\in\mathcal{Q}_t} \left(1 - \frac{1}{B_i}\sum_{s<t} n_{i,s}\right)\widetilde{\mathcal{R}}_{i,t}\right] \quad \text{by Theorem 2.2 (i)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{2} \sum_{i\in[N]} \sum_{t\in[T]} \left(1 - \frac{1}{B_i} \sum_{s<t} \mathbb{E}\left[n_{i,s}\right]\right) \widetilde{\mathcal{R}}_{i,t} \\
&\geq \frac{1}{2} \sum_{i\in[N]} \sum_{t\in[T]} \left(1 - \frac{1}{B_i} \sum_{s<t} \widetilde{\mathcal{K}}_{i,s}\right) \widetilde{\mathcal{R}}_{i,t} \text{ by (A.2)} \\
&\geq \frac{1}{2} \sum_{i\in[N]} (\sqrt{2}-1) \sum_{t\in[T]} \mathcal{R}_{i,t} \text{ by Theorem 2.3} \\
&= \frac{1}{2}(\sqrt{2}-1) \sum_{i\in[N]} \sum_{t\in[T]} \mathcal{R}_{i,t} \\
&\geq \frac{1}{2}(\sqrt{2}-1)\ \mathsf{OPT} \text{ by Theorem 2.1.}
\end{aligned}
$$

□

A.5 Proof of Proposition 2.1

Proof. Consider an algorithm that uses Framework. For each arm m, let n_m be the number of times arm m is pulled by the algorithm. Recall that $K_m := \min\{U_m, W\}$. Let W_m be the remaining budget of budget W when the algorithm starts to pull arm m, and define $K'_m := \min\{U_m, W_m\}$. Recall $\mathcal{R}'_m := \mathcal{R}(\mathcal{P}_m(\cdot, K_m))$ an $\mathcal{K}'_m := \mathcal{K}(\mathcal{P}_m(\cdot, K_m))$.

By Theorem 2.2 (i), we have

$$\mathcal{R}(\mathcal{P}_m(\cdot, K'_m)) \geq \frac{K'_m}{K_m} \cdot \mathcal{R}'_m = \frac{\min\{U_m, W_m\}}{\min\{U_m, W\}} \cdot \mathcal{R}'_m.$$

Given order $I(1), \ldots, I(M)$, which is a permutation of $1, \ldots, M$, we know $W_{I(k)} = W - \sum_{j<k} n_{I(j)}$.

Hence we have

$$\frac{\min\{U_{I(k)}, W_{I(k)}\}}{\min\{U_{I(k)}, W\}} \geq \frac{W_{I(k)}}{W} \geq \frac{1}{W}\left(W - \sum_{j<k} n_{I(j)}\right)$$

Therefore, we know

$$\mathcal{R}(\mathcal{P}_{I(k)}(\cdot, K'_{I(k)})) \geq \frac{1}{W}\left(W - \sum_{j<k} n_{I(j)}\right)\mathcal{R}'_{I(k)}.$$

By Theorem 2.2 (ii), we also know

$$\mathbb{E}\left[n_{I(j)}\right] \leq \mathcal{K}'_{I(j)}.$$

Let ALG be the expected reward of the algorithm, we have

$$\begin{aligned}
\mathsf{ALG} &= \mathbb{E}\left[\sum_{k\in[M]} \mathcal{R}(\mathcal{P}_{I(k)}(\cdot, K'_{I(k)}))\right] \\
&\geq \mathbb{E}\left[\sum_{k\in[M]} \left(1 - \frac{1}{W}\sum_{j<k} n_{I(j)}\right)\mathcal{R}'_{I(k)}\right] \text{ (by Theorem 2.2 (i))} \\
&= \sum_{k\in[M]} \left(1 - \frac{1}{W}\sum_{j<k} \mathbb{E}\left[n_{I(j)}\right]\right)\mathcal{R}'_{I(k)} \\
&\geq \sum_{k\in[M]} \left(1 - \frac{1}{W}\sum_{j<k} \mathcal{K}'_{I(j)}\right)\mathcal{R}'_{I(k)} \text{ (by Theorem 2.2 (ii))}.
\end{aligned}$$

□

A.6 Proof of Proposition 2.2

Proof. To prove towards a contradiction, suppose the lower bound is minimized when (2.14) is not satisfied. Then we know there must exist two consecutive indices s and t with $0 \leq s < t \leq M$ such that

$$\frac{\mathcal{R}'_{I(s)}}{\mathcal{K}'_{I(s)}} > \frac{\mathcal{R}'_{I(t)}}{\mathcal{K}'_{I(t)}}.$$

Recall the lower bound in (2.12) is

$$\sum_{k\in[M]}\left(1-\frac{1}{W}\sum_{j<k}\mathcal{K}'_{I(j)}\right)\mathcal{R}'_{I(k)} \tag{A.3}$$

Now consider switching the positions of arms $I(s)$ and $I(t)$. Let Δ denote the change in the lower bound after this switch. We have

$$\begin{aligned}\Delta &= \left(1-\frac{1}{W}\sum_{j<s}\mathcal{K}'_{I(j)}\right)\mathcal{R}'_{I(t)} + \left(1-\frac{1}{W}\sum_{j<s}\mathcal{K}'_{I(j)}-\frac{1}{W}\mathcal{K}'_{I(t)}\right)\mathcal{R}'_{I(s)}\\ &\quad - \left(1-\frac{1}{W}\sum_{j<s}\mathcal{K}'_{I(j)}\right)\mathcal{R}'_{I(s)} - \left(1-\frac{1}{W}\sum_{j<s}\mathcal{K}'_{I(j)}-\frac{1}{W}\mathcal{K}'_{I(s)}\right)\mathcal{R}'_{I(t)}\\ &= \frac{1}{W}\mathcal{K}'_{I(s)}\mathcal{R}'_{I(t)} - \frac{1}{W}\mathcal{K}'_{I(t)}\mathcal{R}'_{I(s)} < 0.\end{aligned}$$

Therefore, by switching the positions of arms $I(s)$ and $I(t)$, we can further decrease the lower bound, which results in a contradiction.

Note that we assume all single-arm policies $\mathcal{P}_m(\cdot)$ have strictly positive expected costs. If the expected cost of a single arm policy is zero, which implies its expected reward is also zero, then it can be positioned anywhere without changing the lower bound of the expected reward.

□

A.7 Proof of Lemma 2.1

Proof. Let $K^* = \sum_{t\in[T]}\mathcal{K}_t$, and $x_0 := K^*/B$. By definition, we know $F(0) = 0$, $F(x_0) = 1$, and $F(x)$ is strictly increasing in x over $[0, K^*/B]$. Let $G(x) := 1 - F(x)$. Then we know $G(0) = 1$, $G(x_0) = 0$, and $G(x)$ is strictly decreasing in x over $[0, K^*/B]$.

In addition, by the definition of $F'(\cdot)$, we have

$$F'(0) := \left[0, \frac{\mathcal{R}_1/R^*}{\mathcal{K}_1/B}\right] \text{ and } F'(x_0) := \left[\frac{\mathcal{R}_T/R^*}{\mathcal{K}_T/B}, \infty\right),$$

where $\mathcal{R}_1/\mathcal{K}_1$ and $\mathcal{R}_T/\mathcal{K}_T$ are the lowest and highest reward-to-cost ratios after we re-arrange the arms.

Since $\mathcal{R}_T/\mathcal{K}_T \geq R^*/K^*$, we have

$$F'(x_0) \geq \frac{B}{K^*} \geq 1.$$

The cutting point satisfies $G(x) := 1 - F(x) \in F'(x)$. Therefore, by the continuity of $F'(x)$ and the monotonicity of $G(x)$, we know x^* is guaranteed to exist, and $x^* \in [0, x_0]$. The uniqueness of x^* follows from the strict monotonicity of $G(x)$.

□

A.8 Proof of Lemma 2.2

Proof. By definition (2.16), we know the curve of $F(x)$ is convex, and thus the cutting point satisfies the following conditions.

- $y^* - x^*(1 - y^*) \leq 0$
- $0 \leq x^*, y^* \leq 1$

Given condition (2.17), namely, $1 - F(x^*) \in F'(x^*)$, and definition $y^* := F(x^*)$, we know

$$e_1 + e_2 \geq \frac{1}{2}(1 + (x^*)^2)(1 - y^*). \tag{A.4}$$

In addition, the right-hand side of (A.4) is lower-bounded by the value of the optimization problem

$$\begin{aligned} \min_{x,y \in [0,1]} \quad & \frac{1}{2}(1 + x^2)(1 - y) \\ s.t. \quad & y - x(1 - y) \leq 0. \end{aligned}$$

We apply the KKT conditions to solve this optimization problem. Specifically, by the first-order condition, we find a unique stationary point

$$(x', y', \mu') = (\sqrt{2} - 1, 1 - \frac{1}{\sqrt{2}}, \sqrt{2} - 1).$$

The corresponding Hessian matrix is

$$H = \begin{bmatrix} 1 - y & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}.$$

For all $\mathbf{d} := (d_1, d_2) \in T(x', y')$, where $T(x', y') = \{\mathbf{d} : d_1 = 2d_2\}$, and $\mathbf{d} \neq \mathbf{0}$, we have $\mathbf{d}^T H \mathbf{d} > 0$. Hence the second-order condition is also satisfied. Therefore, we have

$$e_1 + e_2 \geq \frac{1}{2}(1 + (x')^2)(1 - y') = \sqrt{2} - 1.$$

□

A.9 Proof of Lemma A.1

Lemma A.1. *Consider an arm and its associated MDP with underlying state space S. The expected reward p^u at state $u \in S$ satisfies the martingale property, namely,*

$$\mathbb{E}\left[p^v \mid p^u\right] = \sum_{v \in S} q^{u,v} p^v = p^u,$$

where $q^{u,v}$ denote the MDP's transition probability from state u to state v.

Proof. Suppose the arm is in state $u = (a, b)$. Let $s \in \{0, 1\}$ denote the binary result after we pull the arm while it is in state u. Let $Pr(\cdot)$ denote the probability function conditional on the value of p^u. Then we have,

$$\mathbb{E}[p^v \mid p^u] = \int_0^1 p' \cdot Pr(s = 1) Pr(p' | s = 1) + p' \cdot Pr(s = 0) Pr(p' | s = 0)\, dp'$$

Notice that for any $s \in \{0,1\}$, we have

$$Pr(p'|s) = \frac{Pr(p',s)}{P(s)}.$$

Therefore, we obtain

$$\begin{aligned}
\mathbb{E}[p^v \mid p^u] &= \int_0^1 p' \cdot Pr(p', s=1) + p' \cdot Pr(p', s=0)\, dp' \\
&= \int_0^1 p' \cdot [Pr(p', s=1) + Pr(p', s=0)]\ dp' \\
&= \int_0^1 p' \cdot Pr(p')\, dp' = p^u.
\end{aligned}$$

□

Appendix B

Technical Results in Chapter 3

B.1 Proof of Lemma 3.1

Note that $\mathcal{R}_s(\gamma)$ represents the optimal revenue from supportive products, given that the expected total sales from core products are γ.

By definition (3.3), we can reformulate $\mathcal{R}_s(\gamma)$ as

$$\begin{aligned} \mathcal{R}_s(\gamma) = \min_{\pi} \quad & x \\ \text{s.t.} \quad & x \geq \gamma \cdot F_2(\pi) + F_1(\pi), \ \forall \ \text{feasible policy } \pi, \end{aligned} \tag{B.1}$$

where π denotes the feasible policy of the subproblem.

Formally, the feasible policy π is defined by the feasible solution to problem (3.3), which specifies the values $p_{N+m} \in \Omega_s$, $p'_{N+m} \in \Omega_{sa}$ and $I_{N+m} \in \{0,1\}$, that satisfy $p_{N+m} > p'_{N+m}, \forall \ m = 1, \dots, M$ and $\sum_{m=1}^{M} I_{N+m} \leq S$. Function $F_1(\pi)$ and $F_2(\pi)$ are defined as

$$\begin{aligned} F_1(\pi) \ &:= \ \sum_{m=1}^{M} \alpha_{N+m}(p_{N+m}) p_{N+m} \\ F_2(\pi) \ &:= \ \sum_{m=1}^{M} I_{N+m} \beta'_{N+m}(p'_{N+m}) p'_{N+m} + (1 - I_{N+m}) \beta_{N+m}(p_{N+m}) p_{N+m}. \end{aligned}$$

Observe that in this reformulation, $F_1(\pi)$ and $F_2(\pi)$ are constants, and since the number of feasible policies is finite, the total number of constraints in (B.1) is also finite. Moreover, the RHS of each constraint is a linearly increasing function of γ. Hence we know that for any γ, the optimal solution x is equal to

$$\max_{\pi} \{\gamma \cdot F_2(\pi) + F_1(\pi)\},$$

and we obtain

$$\mathcal{R}_s(\gamma) = \min \max_{\pi} \{\gamma \cdot F_2(\pi) + F_1(\pi)\}.$$

Therefore, we know that $\mathcal{R}_s(\gamma)$ is a convex piece-wise linear function, and it implies that $\mathcal{R}_s(\gamma)$ is Lipschitz continuous. Specifically, the Lipschitz parameter is equal to the function's maximum *slope*, i.e., $\max_\pi F_2(\pi)$, which is bounded by $M \cdot \hat{p}$, by definition.

B.2 Proof of Lemma 3.3

Proof. Let $V(\pi)$ be the *true* revenue of policy π, and $V'(\pi)$ the *approximate* revenue of policy π that is provided by Algorithm 2. In addition, let OPT be the optimal policy of problem (3.1), and ALG the "optimal" policy that is provided by Algorithm 2.

Given policy π, we know that $V(\pi)$ and $V'(\pi)$ give the same revenue for the core products, but different revenue for the supportive products. Specifically, due to the rounding procedure, the value of γ we use in Algorithm 2 differs from its *true* value by at most $N/2K$. Hence by Lemma 3.1, we have

$$|V(\pi) - V'(\pi)| \leq \frac{\hat{p}MN}{2K}, \tag{B.2}$$

for any feasible policy π.

Therefore, we have

$$V(OPT) \leq V^{'}(OPT) + \frac{\hat{p}MN}{2K}$$

$$\leq V'(ALG) + \frac{\hat{p}MN}{2K}$$
$$\leq V(ALG) + \frac{\hat{p}MN}{K},$$

where the first and last inequality follow (B.2). The second inequality follows because ALG optimizes the approximate revenue $V'(\cdot)$.

B.3 Proof of Lemma 3.4

Proof. By definition, the associated counters for primary demand, i.e., $c_n(\cdot)$ and $c_{N+m}(\cdot)$ always increase by 1 in each period, and the associated counters for add-on purchases, i.e., $c_{N+m}^{(a')}(\cdot)$ and $c_{N+m}^{(a)}(\cdot)$ could increase $0, 1, \ldots, N$ in each period, which depends on the total number of core products purchased in the period. In addition, to obtain an upper bound on the length of an episode, it suffices to consider only one counter that is associated with the primary demand. W.L.O.G., consider the counter for product 1 with its price determined at the beginning of episode τ, namely, $t(\tau)$. Then we know that the value of this counter is at most $t(\tau)-1$, and the value will be doubled after another $t(\tau)-1$ periods. By the description of Algorithm 3, episode τ starts in period $t(\tau)$ and terminates in no more than $t(\tau)-1$ periods. Therefore, we have $\mathbf{E}[\ell(\tau)] \leq t(\tau)$.

B.4 Proof of Lemma 3.5

Proof. By definition, $\mathcal{E}_t$ is the union of a collection of events.

Specifically, for each core product $n \in \{1, \ldots, N\}$ and price $p_n \in \Omega_c$, given the value of t and $c_{n,t}(p_n)$, by the Chernoff-Hoeffding inequality, we have

$$\mathbf{P}\left\{|\overline{\alpha}_{n,t}(p_n) - \alpha_n(p_n)| > \frac{2\log t}{c_{n,t}(p_n)}\right\} \leq \frac{2}{t^4}.$$

Take the union for all possible values of $c_{n,t}(p_n)$ from 1 to t. By the union bound, we obtain

$$\mathbf{P}\left\{|\overline{\alpha}_{n,t}(p_n)-\alpha_n(p_n)|>\frac{2\log t}{c_{n,t}(p_n)}\right\}\le\frac{2}{t^3}.$$

Similarly, given the value of t, for each supportive product $m\in\{1,\ldots,M\}$ and price $p_{N+m}\in\Omega_s$, take the union for all possible values of $c_{N+m,t}(p_{N+m})$ from 1 to t, and we obtain

$$\mathbf{P}\left\{|\overline{\alpha}_{N+m,t}(p_{N+m})-\alpha_{N+m}(p_{N+m})|>\frac{2\log t}{c_{N+m,t}(p_{N+m})}\right\}\le\frac{2}{t^3}.$$

For add-on purchases, the counters $c^{(a,1)}_{N+m,t}(\cdot)$ and $c^{(a,2)}_{N+m,t}(\cdot)$ range from 1 to Nt. Thus, for each supportive product $m\in\{1,\ldots,M\}$, we obtain

$$\mathbf{P}\left\{\left|\overline{\beta}'_{N+m,t}(p'_{N+m})-\beta'_{N+m,t}(p'_{N+m})\right|>\frac{2\log t}{c^{(a,1)}_{N+m,t}(p'_{N+m})}\right\}\le\frac{2N}{t^3},$$

for each add-on discount price $p'_{N+m}\in\Omega_a$, and

$$\mathbf{P}\left\{\left|\overline{\beta}_{N+m,t}(p_{N+m})-\beta_{N+m}(p_{N+m})\right|>\frac{2\log t}{c^{(a,2)}_{N+m,t}(p_{N+m})}\right\}\le\frac{2N}{t^3},$$

for each add-on original price $p_{N+m}\in\Omega_s$.

Take a union of all these events in $\mathcal{E}_t$, we have

$$\mathbf{P}\left[\mathcal{E}_t\right]\le\frac{2(N+M+2MN)U}{t^3}.$$

In addition, conditional on event $\mathcal{E}_{t(\tau)}$, we know that the regret in episode τ is upper-bounded by $\mathcal{R}^*\cdot\ell(\tau)$. Therefore, for each term on the LHS of (3.8), we have

$$\begin{aligned}&\mathbf{E}\left[(\mathcal{R}^*-\mathcal{R}(\Pi_\tau))\cdot\ell(\tau)\mid\mathcal{E}_{t(\tau)}\right]\cdot\mathbf{P}\left[\mathcal{E}_{t(\tau)}\right]\\ \le\ &\mathcal{R}^*\cdot\ell(\tau)\cdot\frac{2(N+M+2MN)U}{t(\tau)^3}\\ \le\ &\mathcal{R}^*\cdot\frac{2(N+M+2MN)U}{t(\tau)^2}\\ \le\ &\mathcal{R}^*\cdot\frac{2(N+M+2MN)U}{\tau^2},\end{aligned}$$

where the second inequality follows from Lemma 3.4, and the third inequality follows by $t(\tau) \geq \tau$.

Take the sum over τ from 1 to $n(\tau)$. Since $\sum_{\tau=1}^{n(\tau)} 1/\tau^2 \leq \pi^2/6$, we obtain the upper bound

$$\mathcal{R}^* \cdot \frac{(N+M+2MN)U\pi^2}{3},$$

which concludes the proof of Lemma 3.5.

B.5 Proof of Lemma 3.6

Proof. For each term on the LHS of (3.9), we relax probability $\mathbf{P}\left[\mathcal{E}'_{t(\tau)}\right]$ to 1 as an upper bound. We then need to show that the regret is bounded, conditional on event $\mathcal{E}'_{t(\tau)}$ where the empirical mean of each associated parameter is within its confidence interval.

Let Π^* be the optimal policy, and $U_\tau(\Pi^*)$ the value of the objective function of the optimization problem (3.1) under policy Π^* with UCB input parameters $\widetilde{\alpha}_n(p_n)$, $\widetilde{\alpha}_{N+m}(p_{N+m})$, $\widetilde{\beta}'_{N+m}(p'_{N+m})$ and $\widetilde{\beta}_{N+m}(p_{N+m})$, as defined in Algorithm 3. Let $U_\tau(\Pi_\tau)$ be the value of the objective function of the optimization problem (3.1) under policy Π_τ with the same UCB input parameters.

Since the value of the objective function of (3.1) is increasing in all the parameters. We know that conditional on $\mathcal{E}_{t(\tau)}$, $U_\tau(\Pi^*)$ is an upper bound of $\mathcal{R}^*$, namely, the expected revenue of the optimal policy, and $U_\tau(\Pi_\tau)$ is an upper bound of $\mathcal{R}(\Pi_\tau)$. Therefore, we have

$$\begin{aligned}
&\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right]\right] \\
=\ &\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(\mathcal{R}^* - U_\tau(\Pi^*) + U_\tau(\Pi^*) - U_\tau(\Pi_\tau) + U_\tau(\Pi_\tau) - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right]\right] \\
\leq\ &4\hat{p}MN\varepsilon\sqrt{T} + \mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(U_\tau(\Pi_\tau) - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right]\right]. \qquad \text{(B.3)}
\end{aligned}$$

The inequality follows because $\mathcal{R}^* - U_\tau(\Pi^*) \leq 0$ and $U_\tau(\Pi^*) - U_\tau(\Pi_\tau)$ is upper bounded by the approximation error of the FPTAS optimization subroutine. Specifically, with parameter $K = \lceil \sqrt{t(\tau)}\varepsilon \rceil$, we have

$$U_\tau(\Pi^*) - U_\tau(\Pi_\tau) \leq U_\tau(\Pi^*_\tau) - U_\tau(\Pi_\tau) = \frac{\hat{p}MN}{\lceil \sqrt{t(\tau)}/\varepsilon \rceil} \leq \hat{p}MN\varepsilon/\sqrt{t(\tau)}.$$

By Lemma 3.4, we know $\ell(\tau) \leq t(\tau)$. Therefore, we have

$$\sum_{\tau=1}^{n(\tau)} \hat{p}MN\varepsilon/\sqrt{t(\tau)} \leq \sum_{t=1}^{T} \hat{p}MN\varepsilon \cdot \left(2/\sqrt{t}\right) \leq 4\hat{p}MN\varepsilon\sqrt{T}.$$

The second term in (B.3), namely, $\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \mathbf{E}\left[(U_\tau(\Pi_\tau) - \mathcal{R}(\Pi_\tau)) \cdot \ell(\tau) \mid \mathcal{E}'_{t(\tau)}\right]\right]$, describes the *confidence* bound for the revenue of policy Π_τ,

Let $p_{n,\tau}$, $p_{N+m,\tau}$, $p'_{N+m,\tau}$ and $I_{N+m,\tau}$ be the decisions of policy Π_τ.

Conditional on event $\mathcal{E}'_{t(\tau)}$, we obtain

$$U_\tau(\Pi_\tau) - \mathcal{R}(\Pi_\tau) \leq 2(M+1)\hat{p} \cdot \sum_{n=1}^{N} \sqrt{\frac{2\log t(\tau)}{c_{n,t(\tau)}(p_{n,\tau})}} + 2\hat{p} \cdot \sum_{m=1}^{M} \sqrt{\frac{2\log t(\tau)}{c_{N+m,t(\tau)}(p_{N+m,\tau})}}$$
$$+2N\hat{p} \cdot \sum_{m=1}^{M} \left[I_{N+m,\tau} \cdot \sqrt{\frac{2\log t(\tau)}{c^{(a,1)}_{N+m,t(\tau)}(p'_{N+m,\tau})}} + (1 - I_{N+m,\tau}) \cdot \sqrt{\frac{2\log t(\tau)}{c^{(a,2)}_{N+m,t(\tau)}(p_{N+m,\tau})}} \right] \text{(B.4)}$$

To show the inequality, we observe that each of the four parts in (B.4) in fact corresponds to the revenue gap due to the over estimation of the associated demand parameters. In addition, for each parameter, we know that the gap between its true mean and its UCB term is $2\sqrt{\frac{2\log t(\tau)}{\text{counter}}}$ conditional on event $\mathcal{E}'_{t(\tau)}$.

Specifically, we have parameter $\alpha_n(p_{n,\tau})$ contributes to the revenue gap by at most

$$(M+1)\hat{p} \cdot 2\sqrt{\frac{2\log t(\tau)}{c_{n,t(\tau)}(p_{n,\tau})}},$$

parameter $\alpha_{N+m}(p_{N+m,\tau})$ contributes to the revenue gap by at most

$$\hat{p} \cdot 2\sqrt{\frac{2\log t(\tau)}{c_{N+m,t(\tau)}(p_{N+m,\tau})}},$$

parameter $\beta'_{N+m}(p'_{N+m,\tau})$ contributes to the revenue gap by at most

$$N\hat{p} \cdot 2\sqrt{\frac{2\log t(\tau)}{c^{(a,1)}_{N+m,t(\tau)}(p'_{N+m,\tau})}}$$

if $I_{N+m} = 1$, and nothing if $I_{N+m} = 0$; parameter $\beta_{N+m}(p_{N+m,\tau})$ contributes to the revenue gap by at most

$$N\hat{p} \cdot 2\sqrt{\frac{2\log t(\tau)}{c^{(a,2)}_{N+m,t(\tau)}(p_{N+m,\tau})}}$$

if $I_{N+m} = 0$, and nothing if $I_{N+m} = 1$.

Given inequality (B.4), we take a sum over τ on both sides and obtain the following. For parameter $\alpha_n(\cdot)$, we have

$$\begin{aligned}
& \mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \ell(\tau) \cdot \sqrt{\frac{2\log t(\tau)}{c_{n,t(\tau)}(p_{n,\tau})}} \mid \mathcal{E}'_{t(\tau)}\right] \leq \mathbf{E}\left[\sum_{t=1}^{T} \sqrt{\frac{4\log t(\tau)}{c_{n,t}(p_{n,t})}} \mid \mathcal{E}'_{t(\tau)}\right] \\
\leq \; & \mathbf{E}\left[\sum_{i=1}^{\zeta(c)} \sum_{j=1}^{c_{n,T}(q^i_c)} \sqrt{\frac{4\log T}{j}} \mid \mathcal{E}_{t(\tau)'}\right] = 2\sqrt{\log T}\,\mathbf{E}\left[\sum_{i=1}^{\zeta(c)} \sum_{j=1}^{c_{n,T}(q^i_c)} \sqrt{\frac{1}{j}} \mid \mathcal{E}'_{t(\tau)}\right] \\
\leq \; & 2\sqrt{\log T}\, \left[\sum_{i=1}^{\zeta(c)} 2\sqrt{c_{n,T}(q^i_c)} \mid \mathcal{E}'_{t(\tau)}\right] \leq 2\sqrt{\log T}\,\mathbf{E}\left[2\sqrt{UT} \mid \mathcal{E}'_{t(\tau)}\right] \\
= \; & 4\sqrt{UT\log T}. \qquad \text{(B.5)}
\end{aligned}$$

In the first inequality, with abuse of notation, we use $p_{n,t}$ to denote the price decision of product n in period t. The inequality follows due to the fact that the counter's value in period τ is no larger than $2 \cdot t(\tau)$. The second inequality follows from the relaxation of $\log t(\tau)$ to $\log T$, and an alternative way of counting $c_{n,t}(p_{n,t})$ from $t = 1$ to T. The third inequality follows because of the fact that $\sum_{j=1}^{J} \sqrt{1/j} \leq 2\sqrt{J}$. The last inequality follows from the Cauchy-Schwarz inequality since we know

$\zeta(c) := |\Omega_c| \leq U$ and $\sum_{i=1}^{\zeta(c)} c_{n,T}(q_c^i) \leq T$.

Following the same analysis, we can show similar bounds for parameters $\alpha_{N+m}(\cdot)$, $\beta'_{N+m}(\cdot)$ and $\beta_{N+m}(\cdot)$. Notice that in developing the bounds for $\beta'_{N+m}(\cdot)$ and $\beta_{N+m}(\cdot)$, we need to have the values of the associated counters $c_{N+m,T}^{(a,1)}(\cdot)$ and $c_{N+m,T}^{(a,2)}(\cdot)$ to be non-zero for at least one of the prices. Hence, we need to multiple the bound by $1/\lambda$, where λ denotes the lowest probability that the total primary demand is non-zero.

Now given (B.4) and (B.5), we have

$$\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \ell(\tau) \cdot (U_\tau(\Pi_\tau) - \mathcal{R}(\Pi_\tau)] \mid \mathcal{E}_{t(\tau)}\right] \leq \left[2(M+1)N + 2M + \frac{4NM}{\lambda}\right] \hat{p} \cdot 4\sqrt{UT \log T}.$$

Put the bound back to (B.3), and we obtain

$$\mathbf{E}\left[\mathbf{E}\left[\sum_{\tau=1}^{n(\tau)} \ell(\tau) \cdot (\mathcal{R}^* - \mathcal{R}(\Pi_\tau)] \mid \mathcal{E}'_{t(\tau)}\right] \cdot \mathbf{P}\left[\mathcal{E}'_{t(\tau)}\right]\right] \leq 4\hat{p}MN\varepsilon\sqrt{T} + \frac{8NM}{\lambda}\hat{p} \cdot 4\sqrt{UT \log T}.$$

This concludes the proof of Lemma 3.6.

Appendix C

Technical Results in Chapter 4

C.1 Efficient Projection Algorithms

We use the following algorithm to project $\lambda(t) \notin \Omega$ to $\Omega = \{\lambda \mid \lambda \geq 0,\ \|\lambda\|_1 \leq \Lambda\}$ given $\mathbf{v} := \lambda(t)$ and $z := \Lambda$.

Algorithm 8 Efficient Projection to Simplex

```
Input: vector $\mathbf{v}$ and scaler $z > 0$.
    Sort $\mathbf{v}$ into $\mu : \mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$
    Find $\rho = \max\Big\{j \in [n] : \mu_j - \frac{1}{j}\big(\sum_{r=1}^{j} \mu_r - z\big) > 0\Big\}$
    Define $\theta = \frac{1}{\rho}\Big(\sum_{i=1}^{\rho} \mu_i - z\Big)$
Output: $\mathbf{w}$ s.t. $w_i = \max\{v_i - \theta, 0\}$
```

In the following, we present the proof of Theorem 4.2. The proof of Theorem 4.1 simply follows by taking $|\mathcal{X}| = 1$.

C.2 Primal-dual Decomposition

We define the dual benchmark as follows.

$$\mathsf{LP}(\lambda, \theta) : \max_{x} \mathbb{E}_\xi \left[\sum_{k=1}^{K} \left(\sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j a_{ij} \right) d_{ik}(\xi|\theta) \right) x_{\xi,k} \right] + \sum_{j=1}^{M} \lambda_j \frac{I_j}{T} \tag{C.1}$$

$$s.t. \sum_{k=1}^{K} x_{\xi,k} \leq 1, \ \forall \xi \in \mathcal{X} \tag{C.2}$$

$$x_{\xi,k} \geq 0, \ \forall \xi \in \mathcal{X}, \ \forall k \in [K]. \tag{C.3}$$

We denote the optimal value of $\mathsf{LP}(\lambda, \theta)$ as $\mathsf{OPT}(\lambda, \theta)$, where $\lambda = (\lambda_1, \ldots, \lambda_M)$, and the optimal solution to $\mathsf{LP}(\lambda, \theta)$ as $x^*(\lambda, \theta)$.

By duality, we have for any (feasible) $\lambda \geq 0$ that

$$\mathsf{OPT}(\theta) \leq \mathsf{OPT}(\lambda, \theta), \tag{C.4}$$

where $\mathsf{OPT}(\lambda, \theta)$ is given by the optimal value of $\mathsf{LP}(\lambda, \theta)$, i.e.,

$$\mathbb{E}_\xi \left[\sum_{k=1}^{K} \left(\sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j a_{ij} \right) d_{ik}(\xi|\theta) \right) x^*_{\xi,k}(\lambda, \theta) \right] + \sum_{j=1}^{M} \lambda_j \frac{I_j}{T}. \tag{C.5}$$

Given $\xi(t)$ and $\lambda(t)$, define $\mathsf{OPT}(\xi(t), \lambda(t), \theta)$ as

$$\sum_{k=1}^{K} \left(\sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j(t) a_{ij} \right) d_{ik}(\xi(t)|\theta) \right) x^*_{\xi(t),k}(\lambda(t), \theta) + \sum_{j=1}^{M} \lambda_j(t) \frac{I_j}{T}. \tag{C.6}$$

Since $\lambda(t)$ is deterministic given $\mathcal{H}_{t-1}$, we have from (C.4) that

$$\mathsf{OPT}(\theta) \leq \mathbb{E}_{\xi(t)} \left[\mathsf{OPT}(\xi(t), \lambda(t), \theta) \mid \mathcal{H}_{t-1} \right]. \tag{C.7}$$

Suppose the algorithm stops at period τ. We multiply both sides by $\mathbf{1}(\tau \geq t)$, take sum over $t = 1, \ldots, T$, and then take expectation over τ, θ and all randomizations within the algorithm. We have

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}(\tau \geq t) \mathsf{OPT}(\theta) \right] \leq \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}(\tau \geq t) \mathbb{E}\left[\mathsf{OPT}(\xi(t), \lambda(t), \theta) \mid \mathcal{H}_{t-1} \right] \right] \tag{C.8}$$

$$= \mathbb{E}\left[\sum_{t=1}^{T} \mathbb{E}\left[\mathbf{1}(\tau \geq t) \mathsf{OPT}(\xi(t), \lambda(t), \theta) \mid \mathcal{H}_{t-1} \right] \right] \tag{C.9}$$

$$= \mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}\left(\tau \geq t\right) \mathsf{OPT}(\xi(t), \lambda(t), \theta)\right] \tag{C.10}$$

$$= \mathbb{E}\left[\sum_{t=1}^{\tau} \mathsf{OPT}(\xi(t), \lambda(t), \theta)\right]. \tag{C.11}$$

Note that the first equality follows since $\mathbf{1}\left(\tau \geq t\right)$ is deterministic given $\mathcal{H}_{t-1}$.

Let $x^*_{\xi(t),k} := x^*_{\xi(t),k}(\lambda(t), \theta)$ and $x_{\xi(t),k} := \mathbf{1}(k = k(t))$. Let $D_i(t)$ denote the random demand realization, which follows the probability distribution with CDF $F_i(x_1, \ldots, x_N; p_k, \xi(t), \theta)$ and mean $d_{ik}(\xi|\theta)$. We can decompose $\mathsf{OPT}(\xi(t), \lambda(t), \theta)$ by

$$\mathsf{OPT}(\xi(t), \lambda(t), \theta) = R(t, \theta) + \Delta_1(t, \theta) + \Delta_2(t, \theta) + \Delta_3(t, \theta), \tag{C.12}$$

where we have

$$R(t, \theta) := \sum_{k=1}^{K} \left(\sum_{i=1}^{N} p_{ik} d_{ik}(\xi(t)|\theta)\right) x_{\xi(t),k} \tag{C.13}$$

$$\Delta_1(t, \theta) := \sum_{k=1}^{K} \left(\sum_{i=1}^{N} \left(p_{ik} - \sum_{j=1}^{M} \lambda_j(t) a_{ij}\right) d_{ik}(\xi(t)|\theta)\right) \left(x^*_{\xi(t),k} - x_{\xi(t),k}\right) \tag{C.14}$$

$$\Delta_2(t, \theta) := \sum_{j=1}^{M} \left(\frac{I_j}{T} - \sum_{k=1}^{K} \left(\sum_{i=1}^{N} a_{ij} D_i(t)\right) x_{\xi(t),k}\right) \lambda_j(t) \tag{C.15}$$

$$\Delta_3(t, \theta) := \sum_{j=1}^{M} \left(\sum_{k=1}^{K} \left(\sum_{i=1}^{N} a_{ij} \left(D_i(t) - d_{ik}(\xi(t)|\theta)\right)\right) x_{\xi(t),k}\right) \lambda_j(t). \tag{C.16}$$

Therefore, from (C.8) we obtain

$$\mathbb{E}\left[\sum_{t=1}^{T} \mathbf{1}\left(\tau \geq t\right) \mathsf{OPT}(\theta)\right] \leq \mathbb{E}\left[\sum_{t=1}^{\tau} R(t, \theta) + \Delta_1(t, \theta) + \Delta_2(t, \theta) + \Delta_3(t, \theta)\right]. \tag{C.17}$$

C.3 Proof of Proposition 4.1

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \Delta_1(t, \theta)\right] \leq 18 r_{\max} \sqrt{|\mathcal{X}| K T \log K} \tag{C.18}$$

$$\Delta_1(t,\theta) = \sum_{k=1}^{K}\left(\sum_{i=1}^{N}\left(p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right)d_{ik}(\xi(t)|\theta)\right)\left(x^*_{\xi(t),k} - x_{\xi(t),k}\right) \tag{C.19}$$

Proof. We follow the analysis in Ferreira et al. (2018) (Theorem 4 Part I).

Given price $k \in [K]$, context $\xi \in \mathcal{X}$ and dual value λ, let

$$r_{\xi,k}(\lambda) := \sum_{i=1}^{N}\left(p_{ik} - \sum_{j=1}^{M}\lambda_j a_{ij}\right)d_{ik}(\xi|\theta) \tag{C.20}$$

denote the mean *pseudo* revenue.

Recall $\mathcal{H}_{t-1} := (\xi(1), P(1), D(1), \ldots, \xi(t-1), P(t-1), D(t-1))$, i.e., the history available at the beginning of period t. Since parameter $\theta(t)$ is sampled from the posterior distribution given history $\mathcal{H}_{t-1}$, we know $\mathbb{P}(\theta|\mathcal{H}_{t-1}) = \mathbb{P}(\theta(t)|\mathcal{H}_{t-1})$.

Since $\xi(t)$ is sampled i.i.d. from a known distribution, and is independent of θ, $\theta(t)$ and $\mathcal{H}_{t-1}$, we know $\mathbb{P}(\theta|\mathcal{H}_{t-1},\xi(t)) = \mathbb{P}(\theta(t)|\mathcal{H}_{t-1},\xi(t))$. Therefore, we have for $t = 1,\ldots,\tau$ that

$$\mathbb{P}(x^*_{\xi(t),k}|\mathcal{H}_{t-1},\xi(t)) = \mathbb{P}(x_{\xi(t),k}|\mathcal{H}_{t-1},\xi(t)).$$

Let $U_{\xi,k,i}(t)$ be upper confidence bound, which is deterministic given $\mathcal{H}_{t-1}$, namely,

$$U_{\xi,k,i}(t) = \min\left(\bar{d}_i, \hat{d}_{ik\xi}(t-1) + \bar{d}_i\sqrt{\frac{\log_+\left(\frac{TK}{N_{\xi,k}(t-1)\cdot|\mathcal{X}|}\right)}{N_{\xi,k}(t-1)}}\right) \tag{C.21}$$

where

- $\log_+(x) = \log(x)\cdot\mathbf{1}(x \geq 1)$;
- $\bar{d}_i$ denotes the upper bound of demand, namely $D_i(t) \in [0,\bar{d}_i]$;
- $N_{\xi,k}(t-1)$ denotes the number of times when price p_k is offered with context ξ over the first $t-1$ periods;
- $\hat{d}_{ik\xi}(t-1)$ denotes the average demand of product i when price p_k is offered with context ξ in the first $t-1$ periods.

Let $L_{\xi,k,i}(t)$ be lower confidence bound, which is deterministic given $\mathcal{H}_{t-1}$, namely,

$$L_{\xi,k,i}(t) = \max\left(0, \hat{d}_{ik\xi}(t-1) - \bar{d}_i\sqrt{\frac{\log_+\left(\frac{TK}{N_{\xi,k}(t-1)\cdot|\mathcal{X}|}\right)}{N_{\xi,k}(t-1)}}\right) \tag{C.22}$$

Notice that the price term $\left(p_{ik} - \sum_{j=1}^{M}\lambda_j a_{ij}\right)$ in the pseudo revenue could be negative.

To apply the results in Ferreira et al. (2018), we define an adaptive confidence bound $CB_{\xi,k}(t)$, which is deterministic given $\mathcal{H}_{t-1}$, namely,

$$CB_{\xi,k}(t) := \sum_{i=1}^{N} U_{\xi,k,i}(t)\cdot\left(p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right)^+ + L_{\xi,k,i}(t)\cdot\left(\sum_{j=1}^{M}\lambda_j(t)a_{ij} - p_{ik}\right)^+. \tag{C.23}$$

Given $\lambda(t)\in\Omega$, we have for each $k\in[K]$ and $t\in[T]$ that

$$\sum_{i=1}^{N}\left|p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right|\cdot\bar{d}_i \le \sum_{i=1}^{N}p_{ik}\bar{d}_i + \sum_{i=1}^{N}\sum_{j=1}^{M}\lambda_j(t)a_{ij}\bar{d}_i \tag{C.24}$$

$$\le \max_{k\in[K]}\sum_{i=1}^{N}p_{ik}\bar{d}_i + \Lambda\cdot\max_{j\in[M]}\sum_{i=1}^{N}a_{ij}\bar{d}_i. \tag{C.25}$$

The second inequality follows from the fact that $\|fg\|_1 \le \|f\|_1\|g\|_\infty$.

Define the right-hand side of the inequality as $r_{\max}$, namely,

$$r_{\max} := \max_{k\in[K]}\sum_{i=1}^{N}p_{ik}\bar{d}_i + \Lambda\cdot\max_{j\in[M]}\sum_{i=1}^{N}a_{ij}\bar{d}_i. \tag{C.26}$$

For simplicity, we use shorthand notation $r_{\xi(t),k} := r_{\xi(t),k}(\lambda(t))$ and $CB_{\xi(t),k} := CB_{\xi(t),k}(t)$, which are deterministic given $\mathcal{H}_{t-1}$ and $\xi(t)$.

$$\begin{aligned}
&\mathbb{E}\left[\sum_{t=1}^{\tau}\sum_{k=1}^{K}r_{\xi(t),k}\left(x^*_{\xi(t),k} - x_{\xi(t),k}\right)\right] \\
=\ &\mathbb{E}\left[\sum_{t=1}^{\tau}\mathbb{E}\left[\sum_{k=1}^{K}r_{\xi(t),k}x^*_{\xi(t),k} - r_{\xi(t),k}x_{\xi(t),k} \mid \mathcal{H}_{t-1}, \xi(t)\right]\right]
\end{aligned}$$

$$= \mathbb{E}\left[\sum_{t=1}^{\tau}\mathbb{E}\left[\sum_{k=1}^{K} r_{\xi(t),k}x^*_{\xi(t),k} - CB_{\xi(t),k}x^*_{\xi(t),k} + CB_{\xi(t),k}x_{\xi(t),k} - r_{\xi(t),k}x_{\xi(t),k} \mid \mathcal{H}_{t-1}, \xi(t)\right]\right]$$

$$= \mathbb{E}\left[\sum_{t=1}^{\tau}\mathbb{E}\left[\sum_{k=1}^{K} r_{\xi(t),k}x^*_{\xi(t),k} - CB_{\xi(t),k}x^*_{\xi(t),k} \mid \mathcal{H}_{t-1}, \xi(t)\right]\right]$$

$$+ \mathbb{E}\left[\sum_{t=1}^{\tau}\mathbb{E}\left[\sum_{k=1}^{K} CB_{\xi(t),k}x_{\xi(t),k} - r_{\xi(t),k}x_{\xi(t),k} \mid \mathcal{H}_{t-1}, \xi(t)\right]\right]$$

$$= \sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[\mathbf{1}\left(t \geq \tau\right)\left(r_{\xi(t),k}x^*_{\xi(t),k} - CB_{\xi(t),k}x^*_{\xi(t),k}\right)\right] \tag{C.27}$$

$$+ \sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[\mathbf{1}\left(t \geq \tau\right)\left(CB_{\xi(t),k}x_{\xi(t),k} - r_{\xi(t),k}x_{\xi(t),k}\right)\right] \tag{C.28}$$

In term (C.27), since $0 \leq x^*_{\xi(t),k} \leq 1$, we have for all $t \in [T]$, $k \in [K]$ and $\xi(t) \in \mathcal{X}$ that

$$\mathbb{E}\left[r_{\xi(t),k}x^*_{\xi(t),k} - CB_{\xi(t),k}x^*_{\xi(t),k}\right] \leq \mathbb{E}\left[\left(r_{\xi(t),k} - CB_{\xi(t),k}\right)^+\right]. \tag{C.29}$$

In addition, by Lemma EC.3 in Ferreira et al. (2018), which is based on the result in Bubeck Liu (2013) (Theorem 1, Step 2), we have

$$\mathbb{E}\left[\left(r_{\xi(t),k} - CB_{\xi(t),k}\right)^+\right] \leq \sum_{i=1}^{N}\left|p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right| \cdot \bar{d}_i \cdot 6\sqrt{\frac{|\mathcal{X}|}{KT}}\,,$$

which implies

$$\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[r_{\xi(t),k}x^*_{\xi(t),k} - CB_{\xi(t),k}x^*_{\xi(t),k}\right] \leq r_{\max} \cdot 6\sqrt{|\mathcal{X}|KT}. \tag{C.30}$$

For term in (C.28), by Lemma EC.4 in Ferreira et al. (2018), which is based on the results in Bubeck Liu (2013) (Theorem 1, Step 3) and Russo Van Roy (2014) (Proposition 1), we obtain that

$$\sum_{t=1}^{T}\sum_{k=1}^{K}\mathbb{E}\left[CB_{\xi(t),k}x_{\xi(t),k} - r_{\xi(t),k}x_{\xi(t),k}\right] \leq r_{\max} \cdot 12\sqrt{|\mathcal{X}|KT\log K}. \tag{C.31}$$

Therefore, we have

$$\mathbb{E}\left[\sum_{t=1}^{T}\sum_{k=1}^{K} r_{\xi(t),k}\left(x^*_{\xi(t),k} - x_{\xi(t),k}\right)\right] \leq r_{\max}\cdot 18\sqrt{|\mathcal{X}|KT\log K}. \tag{C.32}$$

C.4 Proof of Proposition 4.2

$$\mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_2(t,\theta)\right] \leq \mathbb{E}\left[(\tau - T)\,\mathsf{OPT}(\theta)\right] + \frac{3\sqrt{2}}{2}G\Lambda\sqrt{T} \tag{C.33}$$

$$\Delta_2(t,\theta) = \sum_{j=1}^{M}\left(\frac{I_j}{T} - \sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ij}D_i(t)\right)x_{\xi(t),k}\right)\lambda_j(t) \tag{C.34}$$

Proof. We follow the analysis in Agrawal Devanur (2014b), Agrawal Devanur (2016). Recall that

$$k(t) = \arg\max_{k\in[K+1]}\sum_{i=1}^{N}\left(p_{ik} - \sum_{j=1}^{M}\lambda_j(t)a_{ij}\right)d_{ik}\left(\xi(t)|\theta(t)\right).$$

Given price vector $p_{k(t)}$, context $\xi(t)$ and the true demand parameter θ, define the consumption of resource $j \in [M]$ in period t as $b'_j(t,\theta)$, namely,

$$b'_j(t,\theta) := \sum_{i=1}^{N} a_{ij}D_i(t). \tag{C.35}$$

Recall the dual objective function

$$g_t(\lambda,\theta) := \sum_{j=1}^{M}\lambda_j\cdot\left(\frac{I_j}{T} - \sum_{i=1}^{N}a_{ij}D_i(t)\right). \tag{C.36}$$

The objective in the dual space is to minimize $\sum_{t=1}^{\tau} g_t(\lambda,\theta)$ over domain Ω. Recall

$$\Lambda := \frac{I_{\max}}{I_{\min}}\cdot\sum_{j=1}^{M}p^j_{\max} \quad\text{where}\quad p^j_{\max} := \max_{i:a_{ij}\neq 0,k\in[K]}\frac{p_{ik}}{a_{ij}}. \tag{C.37}$$

By definition, we have the property

$$\Lambda \cdot I_{\min} \geq \mathsf{OPT}(\theta) \cdot T. \tag{C.38}$$

Let $\lambda_{\min}$ be the optimal solution to the dual problem, namely

$$\lambda_{\min} = \arg\min_{\lambda \in \Omega} \sum_{t=1}^{\tau} g_t(\lambda, \theta). \tag{C.39}$$

By Theorem 3.1 in Hazan et al. (2016), we have

$$\sum_{t=1}^{\tau} g_t(\lambda(t), \theta) \leq \sum_{t=1}^{\tau} g_t(\lambda_{\min}, \theta) + \frac{3\sqrt{2}}{2} GD\sqrt{\tau}. \tag{C.40}$$

The inequality follows because

- We know $D \leq \sqrt{2}\Lambda$ by definition.
- Given $\bigtriangledown_j g_t(\lambda) = \frac{I_j}{T} - \sum_{i=1}^{N} a_{ij} D_i(t)$, $j \in [M]$, we have

$$G \leq \sqrt{M} \cdot \max_{j \in [M]} \left| \frac{I_j}{T} - \sum_{i=1}^{N} a_{ij} D_i(t) \right| \tag{C.41}$$

$$\leq \sqrt{M} \cdot \max \left\{ \frac{I_{\max}}{T}, \max_{j \in [M]} \sum_{i=1}^{N} a_{ij} \bar{d}_i \right\} \tag{C.42}$$

by the property $\|a\|_2 \leq \sqrt{M} \cdot \|a\|_1$ for any $a \in \mathbb{R}^M$.

We know the algorithm stops with the following two cases:

1. $\tau < T$, which means $\sum_{t=1}^{\tau} b'_j(t, \theta) \geq I_j$ for some j. In this case, since $\Lambda \cdot e_j \in \Omega$, we have

$$\sum_{t=1}^{\tau} g_t(\lambda_{\min}, \theta) \leq \sum_{t=1}^{\tau} g_t(\Lambda e_j, \theta) \leq \Lambda \cdot \left(\tau \cdot \frac{I_j}{T} - I_j \right) \leq \Lambda \cdot \left(\frac{\tau}{T} - 1 \right) \cdot I_{\min}. \tag{C.43}$$

2. $\tau = T$. In this case, we have

$$\sum_{t=1}^{\tau} g_t(\lambda_{\min}, \theta) \leq \sum_{t=1}^{\tau} g_t(0) = \Lambda \cdot \left(\frac{\tau}{T} - 1 \right) \cdot I_{\min}. \tag{C.44}$$

Recall that

$$\Delta_2(t,\theta) = g_t(\lambda(t),\theta) = \sum_{j=1}^{M}\left(\frac{I_j}{T} - \sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ij} D_i(t)\right) x_{\xi(t),k}\right)\lambda_j(t). \tag{C.45}$$

Therefore, from (C.40), (C.43) and (C.44) we obtain

$$\begin{aligned}\sum_{t=1}^{\tau}\Delta_2(t,\theta) &= \Lambda\cdot\sum_{t=1}^{\tau} g_t(\lambda(t),\theta) && \text{(C.46)}\\ &\le \left(\frac{\tau}{T}-1\right)\Lambda I_{\min} + \frac{3\sqrt{2}}{2}G\Lambda\sqrt{T}. && \text{(C.47)}\end{aligned}$$

We have

$$\begin{aligned}\tau\cdot\mathsf{OPT}(\theta) \le{}& \mathbb{E}\left[\sum_{t=1}^{\tau} R(t,\theta)\right] + \mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_1(t,\theta)\right]\\ &+ \left(\frac{\tau}{T}-1\right)\Lambda I_{\min} + \frac{3}{2}G\Lambda\sqrt{T} + \mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_3(t,\theta)\right] \qquad \text{(C.48)}\end{aligned}$$

Given property $\Lambda I_{\min} \ge \mathsf{OPT}(\theta)\cdot T$, since $\tau \le T$, we have

$$\left(\frac{\tau}{T}-1\right)\Lambda I_{\min} \le \tau\cdot\mathsf{OPT} - T\cdot\mathsf{OPT}(\theta). \tag{C.49}$$

Therefore, we obtain

$$T\cdot\mathsf{OPT}(\theta) \le \mathbb{E}\left[\sum_{t=1}^{\tau} R(t,\theta)\right] + \mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_1(t,\theta)\right] + \frac{3}{2}G\Lambda\sqrt{T} + \mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_3(t,\theta)\right] \tag{C.50}$$

C.5 Proof of Proposition 4.3

$$\mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_3(t,\theta)\right] \le q_{\max}\Lambda\left(\sqrt{T\log T}+1\right) \tag{C.51}$$

$$\Delta_3(t,\theta) = \sum_{j=1}^{M}\left(\sum_{k=1}^{K}\left(\sum_{i=1}^{N} a_{ij}\Big(D_i(t) - d_{ik}(\xi(t)|\theta)\Big)\right) x_{\xi(t),k}\right)\lambda_j(t) \tag{C.52}$$

Proof. Construct martingale sequence

$$M(0) := 0, \tag{C.53}$$

$$M(t) := \sum_{s=1}^{t} \sum_{j=1}^{M} \lambda_j(s) \cdot \sum_{i=1}^{N} a_{ij} \Big(D_i(s) - d_{ik(s)}(\xi(s)|\theta) \Big). \tag{C.54}$$

The sequence is a martingale since we have for all $t \in \{1, \dots, \tau\}$,

$$\mathbb{E}[M(t) \mid \mathcal{H}_{t-1}] = M(0) = 0.$$

The regret (stochastic error) is thus

$$\sum_{t=1}^{\tau} \Delta_3(t, \theta) = M(\tau) - M(0) = M(T) - M(0). \tag{C.55}$$

Since $\|fg\|_1 \le \|f\|_1 \|g\|_\infty$ and $\sum_{j=1}^{M} \lambda_j(t) \le \Lambda$, we have

$$|M(t) - M(t-1)| \le \Lambda \cdot \max_{j \in [M]} \sum_{i=1}^{N} a_{ij} \bar{d}_i. \tag{C.56}$$

Define $q_{\max} = \max_j \sum_{i=1}^{N} a_{ij} \bar{d}_i$.

By the Azuma-Hoeffding inequality, we have that with probability at least $1 - \delta$,

$$|M(T) - M(0)| \le \Lambda \cdot q_{\max} \cdot \sqrt{T \log \frac{1}{\delta}}. \tag{C.57}$$

Therefore, we have

$$\mathbb{E}\left[|M(T) - M(0)|\right] \le \Lambda \cdot q_{\max} \cdot \sqrt{T \log \frac{1}{\delta}} \cdot 1 + (T \cdot \Lambda \cdot q_{\max}) \cdot \delta. \tag{C.58}$$

Take $\delta = 1/T$. We obtain

$$\mathbb{E}\left[\sum_{t=1}^{\tau} \Delta_3(t, \theta)\right] \le \mathbb{E}\left[|M(\tau) - M(0)|\right] \le \Lambda \cdot q_{\max} \cdot \left(\sqrt{T \log T} + 1\right) \tag{C.59}$$

C.6 Proof of Theorem 4.3

The proof of Theorem 4.3 follows the decomposition in (C.12). In addition, the regret bounds for $\mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_2(t,\theta)\right]$ and $\mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_3(t,\theta)\right]$ follow the previous analysis. The regret bound for $\mathbb{E}\left[\sum_{t=1}^{\tau}\Delta_1(t,\theta)\right]$ follows from the result of Proposition 3 in Russo Van Roy (2014) where the number of parameters is $d \times N$.

Bibliography

Abbasi-Yadkori Y, Pál D, Szepesvári C (2011) Improved algorithms for linear stochastic bandits. *Advances in Neural Information Processing Systems*, 2312–2320.

Abdallah T (2019) On the benefit (or cost) of large-scale bundling. *Production and Operations Management* 28(4):955–969.

Abdallah T, Asadpour A, Reed J (2017) Large-scale bundle size pricing: A theoretical analysis. *SSRN Electronic Journal* .

Agrawal S, Avadhanula V, Goyal V, Zeevi A (2016a) A near-optimal exploration-exploitation approach for assortment selection. *Proceedings of the 2016 ACM Conference on Economics and Computation*, 599–600 (ACM).

Agrawal S, Avadhanula V, Goyal V, Zeevi A (2017) Thompson sampling for the mnl-bandit. *arXiv preprint arXiv:1706.00977* .

Agrawal S, Avadhanula V, Goyal V, Zeevi A (2019) Mnl-bandit: A dynamic learning approach to assortment selection. *Operations Research* 67(5):1453–1485.

Agrawal S, Devanur NR (2014a) Bandits with concave rewards and convex knapsacks. *Proceedings of the fifteenth ACM conference on Economics and computation*, 989–1006 (ACM).

Agrawal S, Devanur NR (2014b) Fast algorithms for online stochastic convex programming. *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, 1405–1424 (SIAM).

Agrawal S, Devanur NR (2016) Linear contextual bandits with knapsacks. *Advances in Neural Information Processing Systems*, 3450–3458.

Agrawal S, Devanur NR, Li L (2016b) An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. *Conference on Learning Theory*, 4–18.

Agrawal S, Wang Z, Ye Y (2014) A dynamic near-optimal algorithm for online linear programming. *Operations Research* 62(4):876–890.

Auer P, Cesa-Bianchi N, Freund Y, Schapire RE (2002) The nonstochastic multiarmed bandit problem. *SIAM journal on computing* 32(1):48–77.

Badanidiyuru A, Kleinberg R, Slivkins A (2013) Bandits with knapsacks. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 207–216 (IEEE).

Bakos Y, Brynjolfsson E (1999) Bundling information goods: Pricing, profits, and efficiency. *Management science* 45(12):1613–1630.

Ball MO, Queyranne M (2009) Toward robust revenue management: Competitive analysis of online booking. *Operations Research* 57(4):950–963.

Bernstein F, Modaresi S, Sauré D (2018) A dynamic clustering approach to data-driven assortment personalization. *Management Science* 65(5):2095–2115.

Bertsimas D, Niño-Mora J (2000) Restless bandits, linear programming relaxations, and a primal-dual index heuristic. *Operations Research* 48(1):80–90.

Besbes O, Gur Y, Zeevi A (2014) Stochastic multi-armed bandit with non-stationary rewards. *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*.

Besbes O, Gur Y, Zeevi A (2015) Non-stationary stochastic optimization. *Operations research* 63(5):1227–1244.

Besbes O, Zeevi A (2009) Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research* 57(6):1407–1420.

Besbes O, Zeevi A (2012) Blind network revenue management. *Operations research* 60(6):1537–1550.

Besbes O, Zeevi A (2015) On the (surprising) sufficiency of linear models for dynamic pricing with demand learning. *Management Science* 61(4):723–739.

Bubeck S, Cesa-Bianchi N, Lugosi G (2013) Bandits with heavy tail. *IEEE Transactions on Information Theory* 59(11):7711–7717.

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

Bubeck S, Liu CY (2013) Prior-free and prior-dependent regret bounds for thompson sampling. *Advances in Neural Information Processing Systems*, 638–646.

Buchbinder N, Jain K, Naor JS (2007) Online primal-dual algorithms for maximizing ad-auctions revenue. *European Symposium on Algorithms*, 253–264 (Springer).

Cesa-Bianchi N, Lugosi G (2012) Combinatorial bandits. *Journal of Computer and System Sciences* 78(5):1404–1422.

Chen B, Chao X, Ahn HS (2019a) Coordinating pricing and inventory replenishment with nonparametric demand learning. *Operations Research* .

Chen N, Gallego G (2018) A primal-dual learning algorithm for personalized dynamic pricing with an inventory constraint. *Available at SSRN 3301153* .

Chen Q, Jasin S, Duenyas I (2019b) Nonparametric self-adjusting control for joint learning and optimization of multiproduct pricing with finite resource capacity. *Mathematics of Operations Research* 44(2):601–631.

Chen X, Ma W, Simchi-Levi D, Xin L (2019c) Assortment planning for recommendations at checkout under inventory constraints, working paper, Available at SSRN: https://ssrn.com/abstract=2853093.

Chen Y, Shi C (2019) Network revenue management with online inverse batch gradient descent method. *Available at SSRN* .

Cheung WC, Ma W, Simchi-Levi D, Wang X (2018) Inventory balancing with online learning. *arXiv preprint arXiv:1810.05640* .

Cheung WC, Simchi-Levi D (2016) Efficiency and performance guarantees for choice-based network revenue management problems with flexible products. *Available at SSRN 2823339* .

Cheung WC, Simchi-Levi D (2017a) Assortment optimization under unknown multinomial logit choice models. *arXiv preprint arXiv:1704.00108* .

Cheung WC, Simchi-Levi D (2017b) Thompson sampling for online personalized assortment optimization problems with multinomial logit choice models. *Available at SSRN 3075658* .

Cheung WC, Simchi-Levi D, Zhu R (2019) Hedging the drift: Learning to optimize under non-stationarity. *arXiv preprint arXiv:1903.01461* .

Chu CS, Leslie P, Sorensen A, et al. (2011a) Bundle-size pricing as an approximation to mixed bundling. *American Economic Review* 101(1):263.

Chu W, Li L, Reyzin L, Schapire R (2011b) Contextual bandits with linear payoff functions. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 208–214.

Davis J, Gallego G, Topaloglu H (2013) Assortment planning under the multinomial logit model with totally unimodular constraint structures. *Work in Progress* .

Dean BC, Goemans MX, Vondrák J (2008) Approximating the stochastic knapsack problem: The benefit of adaptivity. *Mathematics of Operations Research* 33(4):945–964.

den Boer AV (2015) Dynamic pricing and learning: historical origins, current research, and new directions. *Surveys in operations research and management science* 20(1):1–18.

Devanur NR, Hayes TP (2009) The adwords problem: online keyword matching with budgeted bidders under random permutations. *Proceedings of the 10th ACM conference on Electronic commerce*, 71–78 (ACM).

Duchi J, Shalev-Shwartz S, Singer Y, Chandra T (2008) Efficient projections onto the l 1-ball for learning in high dimensions. *Proceedings of the 25th international conference on Machine learning*, 272–279.

Farias VF, Madan R (2011) The irrevocable multiarmed bandit problem. *Operations Research* 59(2):383–399.

Feldman J, Henzinger M, Korula N, Mirrokni VS, Stein C (2010) Online stochastic packing applied to display ad allocation. *European Symposium on Algorithms*, 182–194 (Springer).

Feldman J, Mehta A, Mirrokni V, Muthukrishnan S (2009) Online stochastic matching: Beating 1-1/e. *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, 117–126 (IEEE).

Feldman JB, Topaloglu H (2017) Revenue management under the markov chain choice model. *Operations Research* 65(5):1322–1342.

Ferreira KJ, Simchi-Levi D, Wang H (2018) Online network revenue management using thompson sampling. *Operations research* 66(6):1586–1602.

Gallego G, Iyengar G, Phillips R, Dubey A (2004) Managing flexible products on a network .

Gallego G, Van Ryzin G (1994) Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science* 40(8):999–1020.

Gallego G, Van Ryzin G (1997) A multiproduct dynamic pricing problem and its applications to network yield management. *Operations research* 45(1):24–41.

Gao X, Jasin S, Najafi S, Zhang H (2018) Multi-product price optimization under a general cascade click model. *Available at SSRN 3262808* .

Gittins J, Glazebrook K, Weber R (2011) *Multi-armed bandit allocation indices* (John Wiley & Sons).

Gittins J, Jones DM (1979) A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika* 66(3):561–565.

Goel G, Mehta A (2008) Online budgeted matching in random input models with applications to adwords. *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, 982–991 (Society for Industrial and Applied Mathematics).

Golrezaei N, Nazerzadeh H, Rusmevichientong P (2014) Real-time optimization of personalized assortments. *Management Science* 60(6):1532–1551.

Guha S, Munagala K (2007) Approximation algorithms for budgeted learning problems. *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, 104–113 (ACM).

Guha S, Munagala K (2008) Sequential design of experiments via linear programming. *arXiv preprint arXiv:0805.2630* .

Guha S, Munagala K (2009) Multi-armed bandits with metric switching costs. *International Colloquium on Automata, Languages, and Programming*, 496–507 (Springer).

Guha S, Munagala K (2013) Approximation algorithms for bayesian multi-armed bandit problems. *arXiv preprint arXiv:1306.3525* .

Guha S, Munagala K, Shi P (2010) Approximation algorithms for restless bandit problems. *Journal of the ACM (JACM)* 58(1):3.

Hazan E, et al. (2016) Introduction to online convex optimization. *Foundations and Trends® in Optimization* 2(3-4):157–325.

Hitt LM, Chen Py (2005) Bundling with customer self-selection: A simple approach to bundling low-marginal-cost goods. *Management Science* 51(10):1481–1493.

Jaillet P, Lu X (2012) Near-optimal online algorithms for dynamic resource allocation problems. *arXiv preprint arXiv:1208.2596* .

Jaillet P, Lu X (2013) Online stochastic matching: New algorithms with better bounds. *Mathematics of Operations Research* 39(3):624–646.

Jin R, Simchi-Levi D, Wang L, Wang X, Yang S (2019) Shrinking the upper confidence bound: A dynamic product selection problem for urban warehouses. *arXiv preprint arXiv:1903.07844* .

Kallus N, Udell M (2016) Dynamic assortment personalization in high dimensions. *arXiv preprint arXiv:1610.05604* .

Kök AG, Fisher ML, Vaidyanathan R (2008) Assortment planning: Review of literature and industry practice. *Retail supply chain management*, 99–153 (Springer).

Lei YM, Jasin S, Sinha A (2014) Near-optimal bisection search for nonparametric dynamic pricing with inventory constraint. *Ross School of Business Paper* (1252).

Liu Q, Van Ryzin G (2008) On the choice-based linear programming model for network revenue management. *Manufacturing & Service Operations Management* 10(2):288–310.

Ma W (2018) Improvements and generalizations of stochastic knapsack and markovian bandits approximation algorithms. *Mathematics of Operations Research* 43(3):789–812.

Ma W, Simchi-Levi D (2015) Reaping the benefits of bundling under high production costs. *arXiv preprint arXiv:1512.02300* .

Ma W, Simchi-Levi D (2019) Algorithms for online matching, assortment, and pricing with tight weight-dependent competitive ratios. *arXiv preprint arXiv:1905.04770* .

Mahdian M, Yan Q (2011) Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. *Proceedings of the forty-third annual ACM symposium on Theory of computing*, 597–606 (ACM).

Mehta A, Panigrahi D (2012) Online matching with stochastic rewards. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, 728–737 (IEEE).

Mehta A, Saberi A, Vazirani U, Vazirani V (2005) Adwords and generalized on-line matching. *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, 264–273 (IEEE).

Mehta A, et al. (2013) Online matching and ad allocation. *Foundations and Trends in Theoretical Computer Science* 8(4):265–368.

Miao S, Chao X (2017) Dynamic joint assortment and pricing optimization with demand learning. *Forthcoming in Manufacturing & Service Operations Management* .

Miao S, Chao X (2019) Fast algorithms for online personalized assortment optimization in a big data regime. *Available at SSRN 3432574* .

Miao S, Chen X, Chao X, Liu J, Zhang Y (2019) Context-based dynamic pricing with online clustering. *arXiv preprint arXiv:1902.06199* .

Nino-Mora J (2001) Restless bandits, partial conservation laws and indexability. *Advances in Applied Probability* 33(1):76–98.

Özer Ö, Ozer O, Phillips R (2012) *The Oxford handbook of pricing management* (Oxford University Press).

Rusmevichientong P, Shen ZJM, Shmoys DB (2010) Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research* 58(6):1666–1680.

Rusmevichientong P, Tsitsiklis JN (2010) Linearly parameterized bandits. *Mathematics of Operations Research* 35(2):395–411.

Russo D, Van Roy B (2014) Learning to optimize via posterior sampling. *Mathematics of Operations Research* 39(4):1221–1243.

Slivkins A (2019) Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272* .

Talluri K, Van Ryzin G (2004) Revenue management under a general discrete choice model of consumer behavior. *Management Science* 50(1):15–33.

Talluri KT, Van Ryzin GJ (2006) *The theory and practice of revenue management*, volume 68 (Springer Science & Business Media).

Thompson WR (1933) On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25(3/4):285–294.

Truong VA (2015) Optimal advance scheduling. *Management Science* 61(7):1584–1597.

Wang Z, Deng S, Ye Y (2014) Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Operations Research* 62(2):318–331.

Whittle P (1980) Multi-armed bandits and the gittins index. *Journal of the Royal Statistical Society: Series B (Methodological)* 42(2):143–149.

Wu Sy, Hitt LM, Chen Py, Anandalingam G (2008) Customized bundle pricing for information goods: A nonlinear mixed-integer programming approach. *Management Science* 54(3):608–622.

Yuan H, Luo Q, Shi C (2019) Marrying stochastic gradient descent with bandits: Learning algorithms for inventory systems with fixed costs, working paper.

Zhang D, Adelman D (2009) An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science* 43(3):381–394.

Zhang H, Chao X, Shi C (2017) Perishable inventory problems: Convexity results for base-stock policies and learning algorithms under censored demand. *Forthcoming in Operations Research* .

Zhang H, Chao X, Shi C (2019) Closing the gap: A learning algorithm for lost-sales inventory systems with lead times. *Management Science* .