Adaptive Optimization and Learning for Service Systems

by

Huiwen Jia

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Industrial and Operations Engineering)
in the University of Michigan
2022

Doctoral Committee:

Associate Professor Siqian Shen, Co-Chair
Associate Professor Cong Shi, Co-Chair
Professor Jon Lee
Professor Yafeng Yin

Huiwen Jia
hwjia@umich.edu
ORCID iD: 0000-0002-2633-9278

© Huiwen Jia 2022

ACKNOWLEDGMENTS

This acknowledgment demonstrates part of my deep appreciations to everyone who walks alongside me and empowers me in this journey.

I would like to express my sincere gratitude to my advisor, Dr. Siqian Shen. I am indeed fortunate that she took a chance to advise me when I was a master student and encouraged me to pursue a Ph.D. degree, which has immeasurably changed my life. Her fabulous inventiveness, tireless passion, and great perseverance taught me how to be a successful and excellent student, scholar, colleague, friend, and person. Among all incredible women leaders who have indelibly impacted my life, I count her chief.

I am also intensely privileged to be co-advised by Dr. Cong Shi. He has always been a source of inspirations and support. His enthusiasm, patience, warmth, conscientiousness, and thoughtfulness are unparalleled. Thank you to Dr. Shi for advising and growing my skills, trusting and valuing my efforts, shaping and advancing my directions, and building and enhancing my confidence. You have taught me more than I could ever give you credit for here.

I greatly appreciate my outstanding preliminary examination and dissertation committee members, Dr. Jon Lee, Dr. Ruiwei Jiang, and Dr. Yafeng Yin, for their precious wisdom and time in providing me helpful feedback and supports. I owe a debt of gratitude to Dr. Chen Wang, Dr. Philip Kaminsky, Dr. Mohsen Moarefdoost, and Mr. Jun Fang for their generous mentorship during my undergraduate study and internships. I have also been deeply benefitted to work closely with Yan Deng, Xian Yu, Xinyu Fei, Kati Moug, and Gongyu Chen on different research topics.

My Ph.D. life have been kept on track with the love, support, and encouragement of many wonderful individuals and groups. A special thanks to Yiling, Miao, Hideaki, Suyanpeng, Luze, Junhong, Weiyu, Na, Haoming, Xubo, Jingwen, Mengxin, Yiwen, Yiyi, Siyu, Yehao, Liyang, Ruowei, Jing, Keyu, Leyi, Lishan, Nan, Ece, Qi, Sentao, Matthew, Kammie, Kevin, and Jorge.

I would not be where I am today without the support of my family. Thank you to my parents, Guanghua Jia and Caixia Han, for their endless love and unwavering support throughout the years. Thank you to my husband, Weikun Yang, for his unconditional support and for giving me the courage and strength to arrive at this milestone. Thank you to my lovely little brothers, Huipu and Huixun, for the sense of honour and responsibility you bring to me. Thank you to my cats, Bear and Lychee, for the irreplaceable companionship and the patience when they wait for the playtime.

TABLE OF CONTENTS

ACKNOWLEDGMENTS ii

LIST OF FIGURES vi

LIST OF TABLES vii

LIST OF ALGORITHMS viii

LIST OF APPENDICES ix

ABSTRACT x

CHAPTER

1 Introduction 1

1.1 Background 1
1.1.1 Benders Decomposition for Solving Two-Stage Stochastic Programs . . . 2
1.1.2 Vehicle Routing in On-demand Ride-Hailing Systems 4
1.1.3 Revenue Management with Reusable Resources 5
1.2 Dissertation Overview 7

2 Benders Cut Classification via Support Vector Machines for Solving Two-stage Stochastic Programs 9

2.1 Introductory Remarks 9
2.1.1 An Overview of Benders Decomposition 10
2.1.2 Challenges and Research Overview 12
2.1.3 Contributions of the Paper 13
2.1.4 Structure of the Paper 13
2.2 Literature Review 14
2.3 Learning-enhanced Benders Decomposition 15
2.3.1 Phase 1: Cut Sampling 16
2.3.2 Subroutine in Phase 2: Classifier Construction 19
2.3.3 Phase 2: Cut Classification 23
2.4 Numerical Studies 25
2.4.1 Experimental Setup and Test Instances 25
2.4.2 Performance of the SVM Classifier 26
2.4.3 Results of Comparing LearnBD with Benders Decomposition 27

2.4.4 Performance Comparison Over Iterations . . . 33
2.4.5 Sampling and Training Time . . . 36
2.4.6 Classifier Transfer Between Instances . . . 36
2.5 Concluding Remarks . . . 38

3 Context-aware Route Recommendation with Weight Learning through Deep Neural Networks . . . 40

3.1 Introductory Remarks . . . 40
3.1.1 WSPP . . . 41
3.1.2 Literature Review of Context-aware Route Planning . . . 42
3.1.3 Contributions . . . 43
3.1.4 Structure . . . 43
3.2 Problem Description and Notations . . . 44
3.3 Learning-Based Context-aware Route Recommendation System (LCRRS) . . . 45
3.4 Computational Studies . . . 48
3.4.1 Data Collection . . . 48
3.4.2 Lower and Upper Bounds of Acceptance Ratio . . . 49
3.4.3 Results with Different Contextual Information . . . 50
3.4.4 Results with Different Implementation Variants . . . 51
3.5 Concluding Remarks . . . 52

4 Online Learning and Pricing for Service Systems with Reusable Resources . . . 53

4.1 Introductory Remarks . . . 54
4.1.1 Main Results and Contributions . . . 55
4.1.2 Literature Review . . . 57
4.1.3 Structure of the Paper . . . 61
4.2 Problem Formulation . . . 61
4.2.1 Regret, Relaxed Regret, and Static Price Benchmark . . . 63
4.3 Online Learning and Pricing Algorithms: BUCB and BTS . . . 66
4.3.1 Preliminaries of UCB and TS . . . 66
4.3.2 Online Batch Upper Confidence Bound Algorithm (BUCB) . . . 67
4.3.3 Online Batch Thompson Sampling Algorithm (BTS) . . . 69
4.4 Performance Analysis of BUCB and BTS . . . 70
4.4.1 High Probability Bound for Mixing Time . . . 71
4.4.2 Performance of BUCB . . . 73
4.4.3 Performance of Online Batch Thompson Sampling (BTS) . . . 77
4.5 Model with Continuous Price Set under Contextual Information . . . 79
4.5.1 Batch LinUCB Algorithm . . . 80
4.5.2 Performance Analysis of BLinUCB . . . 86
4.6 Extension to Settings with Balking and Reneging . . . 90
4.6.1 Regret for Extended Systems with Balking and Reneging . . . 91
4.6.2 BUCB and BTS for Extended Systems with Balking and Reneging . . . 92
4.6.3 Performance Analysis for Systems with Balking and Reneging . . . 96
4.7 Numerical Experiments . . . 97
4.7.1 Numerical Results of BUCB and BTS with Discrete Price Set . . . 97

4.7.2 Numerical Results of BLinUCB with Continuous Price Set 103
4.7.3 Continuous VS Discrete Time Control 107
4.8 Concluding Remarks 107

5 Online Learning and Pricing for Network Service Systems with Reusable Resources . 110

5.1 Introductory Remarks 111
5.1.1 Main Results and Contributions 112
5.1.2 Literature Review 114
5.1.3 Structure of the Paper 117
5.2 Problem Formulation 118
5.2.1 Illustrative Example 120
5.3 Regret, Relaxed Regret and Static Price Benchmark 121
5.4 Algorithms 124
5.4.1 Probabilistic Concentration Bound 125
5.4.2 RNRM-UCB Algorithm 128
5.4.3 RNRM-TS Algorithm 129
5.5 Performance Analysis 129
5.5.1 High Probability Bound for Mixing Time 130
5.5.2 Performance of RNRM-UCB 139
5.5.3 Performance of RNRM-TS 143
5.6 Concluding Remarks 146

6 Conclusion 147

APPENDICES 149

BIBLIOGRAPHY 187

LIST OF FIGURES

FIGURE

1.1 An example of WSPP on $G(N, A)$ 5

2.1 An overview of LearnBD procedures. In Phase 1, we collect cuts from training problems and label them as valuable cuts (green points) and non-valuable cuts (blue points). Then, we train an SVM classifier and use it to classify the new cuts we meet in Phase 2 when solving the testing problem. In Phase 2, we only include the valuable cuts identified by the SVM classifier in each iteration. 17
2.2 CFLP: Problem set IV, cap41 instance solved by BD and LearnBD. The horizontal axis is the iteration number. 34
2.3 CMND: Problem set VII, r075 instance solved by BD and LearnBD. The horizontal axis is the iteration number. 35

3.1 An example of WSPP on $G(N, A)$ 42
3.2 Overview of LCRRS 46

4.1 Roadmap of BUCB and BTS Algorithms and Regret Analysis. 71
4.2 Offered Price and Time-average Cumulative Regret of Policies in Instances #1 to #3 (Rates-prices Scenario 1 Under Three Balking and Reneging Cases). 100
4.3 Offered Price and Time-average Cumulative Regret of Policies in Instances #4 to #6 (Rates-prices Scenario 2 Under Three Balking and Reneging Cases). 101
4.4 Offered Price and Time-average Cumulative Regret of Policies in Instances #7 to #9 (Rates-prices Scenario 3 Under Three Balking and Reneging Cases). 102
4.5 Function value of $\phi(p)$. 104
4.6 Offered Price and Time-average Cumulative Regret of Policies in Instances #10 to #12. 106
4.7 Offered Price and Time-average Cumulative Regret of Policies in Instances #10, #10.2, and #10.3. 108

5.1 Servers and products in a network system. 121
5.2 Roadmap of RNRM-UCB and RNRM-TS Algorithms and Regret Analysis. 131
5.3 The left part shows the arrival rates of the actual, alternative, and virtual queues (for a Layer-2 node i when the Jackson network starting from empty state). The right part shows the probability that the queue has different number of customers at time τ. This probability are the same for the alternative and the actual queues and thus they are equivalent systems after time τ. 135
5.4 The mixing time analysis for an illustrative example starting from zero state. 136

B.1 Intervals within Busy Period A_n. 163

LIST OF TABLES

TABLE

2.1 Instance Attributes of CFLP 26
2.2 Instance Attributes of CMND 26
2.3 Prediction Accuracy of the SVM Classifier 27
2.4 Results of CFLP Instances Solved by LearnBD and BD 28
2.5 Comparing LearnBD with BD for Solving Instance cap62 to Similar Accuracy 29
2.6 Results of CMND Instances Solved by LearnBD and BD 30
2.7 Comparing LearnBD with BD for Solving CMND instances to Similar Accuracy . . . 31
2.8 Δ-values Throughout Computational Iterations of Five CMND Instances 32
2.9 Results of Multiple Runs for CFLP Instance from Problem Set IV 33
2.10 Sampling and Training Time 37
2.11 Results with Transferred Classifier 38

3.1 Results with Benchmarks. 51
3.2 Results with Different Model Structures. 52

4.1 Parameter Settings and Experimental Design (Discrete Price Set) 98
4.2 Parameter Settings and Experimental Design (Continuous Price Range) 104

B.1 Summary of Major Notation for Problem Formulation 157
B.2 Summary of Major Notation for BUCB and BTS 158
B.3 Summary of Major Notation for Coupling Analysis 158
B.4 Summary of Major Notation for BLinUCB 159

C.1 Summary of Major Notation for Problem Formulation 186

LIST OF ALGORITHMS

ALGORITHM

2.1 Phase 1 of the LearnBD algorithm. 19
2.2 Phase 2 of the LearnBD algorithm. 24
4.1 Online Batch Upper Confidence Bound Algorithm (BUCB). 69
4.2 Online Batch Thompson Sampling Algorithm (BTS). 70
4.3 Online Batch LinUCB Algorithm (BLinUCB). 85
4.4 Extended Online Batch Upper Confidence Bound Algorithm (Extended BUCB). . . . 94
4.5 Extended Online Batch Thompson Sampling Algorithm (Extended BTS). 95
4.6 BZ Benchmark (adopted from Besbes and Zeevi 2012). 99
4.7 ϵ-greedy Benchmark. 105
5.1 Online Batch Upper Confidence Bound or Network Revenue Management with Reusable Resourses (RNRM-UCB). 129
5.2 Online Batch Thompson Sampling for Network Revenue Management with Reusable Resources (RNRM-TS). 130

LIST OF APPENDICES

A Appendix For Chapter 2 . . . . . **149**

B Appendix For Chapter 4 . . . . . **157**

C Appendix For Chapter 5 . . . . . **186**

ABSTRACT

The primary focus of this dissertation is to develop adaptive optimization and learning models and algorithms for decision-making problems under uncertainty arising in service systems. Thanks to the accessibility and analyzability of voluminous data, the uncertainties can be better controlled by adaptively incorporating real-time information. The common theme of the problems in different chapters of this thesis embraces the structures of (i) making adaptive decisions for accomplishing a learning task and (ii) learning the nature of the uncertainty for adaptively optimizing future decisions. Specifically, we apply and innovate machine learning and reinforcement learning techniques, including support vector machines (SVM), deep neural networks (DNN), and multi-armed bandits (MAB), to solve problems that arise in modern service systems, such as transportation, resource sharing/rental services.

A well-designed and reliable route planner is central to a wide range of modern transportation applications, including map navigation and on-demand ride-hailing. In addition to efficiently solving a classic weighted shortest path problem, it is critical to understand users' behaviors and preferences, which are generally unknown to the decision-makers. We consider two cases, providing route recommendations to several travel requests or a single request. Providing route recommendations to a fleet of vehicles under uncertainty always results in large-scale stochastic programs, of which the solving process is time-consuming even with sophisticated decomposition algorithms. Therefore, in Chapter 2, we propose a learning-enhanced Benders decomposition (LearnBD) algorithm to reduce the solving time for two-stage stochastic programs. Specifically, we add a cut classification step in each iteration to selectively generate cuts that are more likely to be valuable cuts. This algorithm can also be used for solving general two-stage stochastic programs. In Chapter 3, we design and implement an approach for providing route recommendations to one origin-destination pair via a combination of the weighted shortest path problem and deep learning with real-world transportation data. We employ a deep neural network to learn feature weights from historical choices of route segments, and this network is trained by minimizing the total weighted costs of actual routes and maximizing those of the non-chosen routes.

Revenue management with reusable resources finds a wide range of service systems in today's economy, such as cloud computing services, ride-hailing services, car/bicycle rental services, project team management, hospitality industry, and leisure. The service-providing firm aims to construct a dynamic pricing control to maximize the expected total revenue. The dynamic nature

of reusable resources being committed over time makes this revenue management problem highly challenging, especially when the firm does not know the mappings between customer demand and the offered prices. The solution to this revenue management problem is to make adaptive pricing decisions based on past sales. In Chapter 4, we formulate a revenue management problem under a single product and single reusable resource setting and propose three new pricing algorithms. In Chapter 5, we formulate a network revenue management problem in a multi-products multi-resource setting and propose two new pricing algorithms. We formulate the revenue management problem as a multi-armed bandits (MAB) problem and develop pricing algorithms by leveraging upper confidence bound (UCB) and Thompson sampling (TS) algorithms. As intermediate steps, we establish high probability bounds for how quickly an M/M/c queue and a Jackson network can reach a steady state after the system parameters change.

Overall, the contributions of this dissertation are threefold. First, we develop a LearnBD algorithm to accelerate the solving process of two-stage stochastic programs, and our numerical results of diverse instances show that LearnBD can achieve better computational performance than transitional Benders decomposition for solving different types of benchmark two-stage stochastic programs considered in the literature. Second, we combine deep learning and the weighted shortest path problem to build a weight learning and context-aware route recommendation system. We use real historical request data that was collected from a ride-hailing company to train and test the proposed recommendation system. The computational results indicate that the route recommendations suggested by this new system can achieve a high acceptance ratio. Third, we are among the first to study price-based revenue management problems with reusable resources under incomplete demand information. We develop new batch MAB algorithms for both single-product and multi-product multi-resource settings. A novel aspect of our proposed algorithmic framework is to bound the mixing time loss during transient states between price changes. We derive theoretical regret bounds that match the lower bound up to a logarithmic factor and demonstrate the numerical performances by experiments over various instances.

CHAPTER 1

Introduction

1.1 Background

The primary focus of this dissertation is to combine adaptive optimization and reinforcement learning for decision-making problems under uncertainty in service systems and related applications. Due to the accessibility and analyzability of large volume of data, the uncertainties can be better controlled by adaptively incorporating the real-time information. We mainly focus on two modern service systems, intelligent transportation and rental services with reusable resources.

A well-designed and reliable route planner is central to a wide range of modern transportation applications, including map navigation and on-demand ride-hailing. Traditional route planning is based on the shortest path problem and aims to find the optimal route under a particular metric, such as the shortest distance or traveling time. However, according to Delling et al. (2011) and Letchner et al. (2006), these recommendations are not preferred in real travels due to the inconsistency between the metric used by the route recommendation platform and the behaviors of the users. If the recommended routes are not consistent with users' preferences, they would be abandoned, leading to low user satisfaction rates and eventually causing users to switch to other service providers. The unknown users' behaviors impair the effectiveness of almost all existing traditional route planning models.

Rental services with reusable resources find a wide range of application industries in today's economy, such as cloud computing services, ride-hailing services, car/bicycle rental services, project team management, hospitality industry, and leisure (see, e.g., Williamson 1992, Talluri and Van Ryzin 2006, Özer et al. 2012). This problem is typically studied as a revenue management problem with reusable resources, where a service-providing firm aims to construct a dynamic pricing control to maximize the expected total revenue. Customers' willingness for using the diverse products is affected by the prices being offered; however, in reality, the decision-maker may not know the random demand distributions and how demand reacts to price changes, especially at the beginning of the planning horizon (see, e.g., Xu and Li 2013, Besbes and Zeevi 2012). Moreover,

the total revenue depends on not only the customer demand but also the available resources for services, i.e., the changing real-time capacity of servers. The unknown demand rates and dynamic nature of reusable resources being committed over time make this revenue management problem highly challenging.

Conceptually, adaptive optimization and learning are mutually reinforcing each other. On the one hand, to better accomplish a learning task, one needs to make adaptive decisions for balancing the trade-off between exploration and exploitation. On the other hand, learning the nature of the uncertainty can help make future decisions adaptively to optimize the objective. Therefore, in this dissertation, we embrace adaptive optimization and learning to develop new decision models and solution approaches for solving the above problems, which improves the efficiency, responsiveness, effectiveness, and tractability of modern service systems. Specifically, we apply machine learning techniques, including support vector machines (SVM), deep neural networks (DNN), and multi-armed bandits (MAB), to solve problems that arise in the aforementioned service systems.

We notice that the operational problems for seeking an optimal expected performance in lots of service systems can be formulated as large-scale two-stage stochastic programs, such as recommending routes to a fleet of vehicles. Therefore, we incorporate support vector machines (SVM) to design a learning-enhanced Bender algorithm to accelerate the solving process of this type of program. Specifically, to generate contextual-aware route recommendations, we design and implement an approach via a combination of the weighted shortest path problem and deep neural networks (DNN) to learn unknown user preferences toward multiple route features using real-world transportation data. We are among the first to study price-based revenue management problems with reusable resources under incomplete demand information. We develop batch MAB type algorithms for both single product and multi-product multi-resource settings.

1.1.1 Benders Decomposition for Solving Two-Stage Stochastic Programs

Benders decomposition (Benders 1962) takes advantage of the decomposable structure of two-stage stochastic programs and is used for solving a broad class of two-stage stochastic programming models (see, e.g., Magnanti and Wong 1981). Consider a general two-stage stochastic programs with complete recourse based on finite samples of the uncertain parameters. The first-stage decision variable $x \in \mathbb{R}^{n_1}$ or $x \in \mathbb{Z}^{n_1}$ is chosen from a feasible region $\mathcal{X}$ before the realization of the uncertainty given the cost vector $c \in \mathbb{R}^{n_1}$. The second-stage decision variable $y \in \mathbb{R}^{n_2}$ is a continuous recourse decision. We consider a finite number of realizations of the uncertain parameter ξ, called "scenarios" or "samples" in the stochastic programming literature (see, e.g., Birge and Louveaux 2011). Set Ω is the sample space that contains all the scenarios and each scenario $\omega \in \Omega$ is associated with a specific realization $\xi_\omega = [W_\omega, h_\omega, T_\omega, q_\omega]$ of the uncertain parameter ξ. Denote

the occurrence probability of scenario ω as p_ω, and thus $\sum_{\omega\in\Omega} p_\omega = 1$. The problem is formulated as:

$$\min_{x\in\mathcal{X}} \quad c^T x + \sum_{\omega\in\Omega} p_\omega Q_\omega(x), \tag{1.1}$$

where

$$\begin{aligned} Q_\omega(x) \stackrel{\text{def}}{=} Q(x,\xi_\omega) = \min_{y} \quad & q_\omega^T y \\ \text{s.t.} \quad & W_\omega y = h_\omega - T_\omega x. \end{aligned} \tag{1.2}$$

The traditional Benders algorithm solves a sequence of master problems and subproblems. By creating new variables $\theta_\omega \in \mathbb{R},\ \forall \omega \in \Omega$ in the first-stage problem, one can formulate a relaxation of the original problem, called relaxed master problem (RMP), which is given by:

$$\begin{aligned} (\text{RMP}^t) \quad \min_{x\in\mathcal{X},\,\theta} \quad & c^T x + \sum_{\omega\in\Omega} p_\omega \theta_\omega \\ \text{s.t.} \quad & \theta_\omega \geq (h_\omega - T_\omega x)^T \nu_\omega \qquad \omega \in \Omega,\ \nu_\omega \in V^{\omega,t}. \end{aligned} \tag{1.3}$$

where $V^{\omega,t}$ is the set of identified extreme points of the feasible region of SP_ω in iteration t (which defined below), and we have $V^{\omega,0} = \emptyset$. Subproblems (SPs) are defined as the linear programming dual of the second-stage problems (1.2) with dual variable $\pi_\omega \in \mathbb{R}^{m_2},\ \forall \omega \in \Omega$. We refer to SP_ω as the SP for scenario ω, formulated as

$$\begin{aligned} (\text{SP}_\omega) \quad Q_\omega^D(x) \ = \max_{\pi_\omega} \quad & (h_\omega - T_\omega x)^T \pi_\omega \\ \text{s.t.} \quad & W_\omega^T \pi_\omega \leq q_\omega. \end{aligned} \tag{1.4}$$

In iteration t, the objective value of RMP^t provides a valid lower bound to the origin problem and $c^T \hat{x}^t + \sum_{\omega\in\Omega} p_\omega Q_\omega(\hat{x}^t)$ provides a valid upper bound. The Benders algorithm terminates when the upper and lower bounds are less than a pre-specified tolerance δ. The algorithm converges in a finite number of iterations due to the finite number of dual extreme rays and extreme points of the finitely many SPs.

While the Benders decomposition helps solve two-stage stochastic programs efficiently, it could suffer from slow convergence. One reason is that the size of RMPs becomes too large due to the quickly increased number of newly added cuts over iterations. Geoffrion and Graves (1974) are among the first to notice and emphasize on the computational difficulty of solving RMPs for stochastic binary integer programs. Magnanti and Wong (1981) report that over 90% of the total time of implementing the Benders decomposition is spent on solving RMPs. Minoux (1986) points out that not all extreme points of the feasible region of SPs equally contribute to restricting the optimal solution to RMPs. Therefore, a larger number of Benders cuts are not tight at the final optimal solution, but can increase the size of RMPs, which are then extremely hard to solve

as large-scale integer programs. Thus, this motivates us to only add valuable cuts to the corresponding RMP in each iteration. Up to date, there is no practical and systematic way to perform cut classification and to accelerate the iterative process for Benders decomposition, according to Rahmaniani et al. (2017). We propose to integrate machine learning techniques into the traditional Benders decomposition framework to learn cut characteristics and selectively generate subsets of Benders cuts iteratively.

1.1.2 Vehicle Routing in On-demand Ride-Hailing Systems

As one fundamental network optimization problem in Geographic Information System (GIS), route planning is essential and of vital importance to a wide range of transportation applications, including map navigation and ride-hailing. For ride-hailing service providers, such as Uber and Didi Chuxing, route planning is a critical business element that provides route recommendations to real-time requests from drivers and riders between certain origin-destination (O-D) pairs. If the recommended routes are not consistent with users' preferences, they would be abandoned, leading to low user satisfaction rates and eventually causing users to switch to other service providers. Therefore, the unknown users' behaviors could impair the effectiveness of almost all existing traditional route planning models.

We typically formulate route planning as a weighted shortest path problem (WSPP). In WSPP, we consider a route evaluation problem where the cost is computed by a weighted sum of multiple route features. Given a route feature set K and a driver preference weight vector w_k for $k \in K$, a route with feature value f_k for $k \in K$ has a cost $\sum_{k \in K} w_k f_k$. As a result, we seek a route with the minimum weighted cost. We construct a directed network $G(N, A)$ from a real-world road map, where nodes in set N represent geographical points and arcs in set A represent road segments that connect nodes in N. Features of arcs, such as travel time and distance, are included in the feature set K. For each feature $k \in K$ of arc $(i, j) \in A$, we evaluate its value f_{ijk}, where smaller value is more preferable, and assign a fixed weight w_k. A travel request is associated with an O-D pair, where $O, D \in N$. Let x_{ij} be a binary decision variable where $x_{ij} = 1$ denotes that arc $(i, j) \in A$ is included in the routing solution, and 0 otherwise. We show an example of WSPP in Figure 1.1.

If w_k is fully known for each feature $k \in K$, we can solve the WSPP through the following

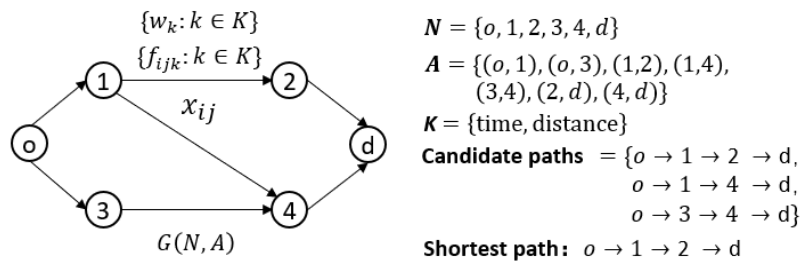


Figure 1.1: An example of WSPP on $G(N, A)$

binary linear program.

$$\text{(WSPP)} \quad \min_{x} \sum_{k \in K} w_k \sum_{(i,j) \in A} f_{ijk} x_{ij} \tag{1.5}$$

$$\text{s.t.} \quad \sum_{j:(o,j) \in A} x_{oj} - \sum_{j:(j,o) \in A} x_{jo} = 1, \tag{1.6}$$

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0, \; i \in N \backslash \{o, d\}, \tag{1.7}$$

$$\sum_{j:(d,j) \in A} x_{dj} - \sum_{j:(j,d) \in A} x_{jd} = -1, \tag{1.8}$$

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in A. \tag{1.9}$$

However, user preferences w_k present substantial heterogeneity and they are not explicitly known when planning routes (see, e.g., Yang et al. (2015)). With advances in the Global Positions System (GPS) and the progress of storing and analyzing the huge volume of human movement data, route recommendation by learning and understanding human activities from historical trajectories becomes a promising area (see, e.g., Cui et al. (2018)). This motivates us to develop a learning-based WSPP evaluation system., which performs three tasks: (i) distinguishing different service contexts; (ii) estimating the weight vectors of drivers; (iii) and providing context-aware route recommendations.

1.1.3 Revenue Management with Reusable Resources

Undoubtedly, revenue management with reusable resources finds a wide range of applications in today's economy. In these applications, the firm is endowed with a finite capacity of reusable products. Each customer requests a product, uses it for some random time, and returns it to the firm, at which point the product can be used to serve other customers (see, e.g., Levi and Radovanović 2010, Chen et al. 2017, Rusmevichientong et al. 2020). For example, firms such as Amazon and Microsoft offer cloud computing services and users utilize virtual machines for a certain duration of time to finish their computational job requests (see, e.g., Kaewpuang et al. 2013, Püschel et al.

2015). The exact usage duration of a request in a cloud is not specified *a priori*, and thus the departure of demand is also stochastic. The firm needs to dynamically decide what price to offer to users based on the current capacity utilization as well as the received requests (see, e.g., Doan et al. 2020). Another salient example arises from the use of on-demand service platforms, such as ride-hailing services including Uber and Lyft, where randomly arrived customers are dynamically paired with service providers (i.e., the resource) and occupy them for a random amount of time (Banerjee et al. 2015, Liu and Li 2017, Bimpikis et al. 2019). The pricing problem in such a two-sided ride-hailing platform also considers assortment simultaneously (see, e.g., Yan et al. 2020). The firm needs to decide what price to charge the riders and how to pay the drivers, as well as dynamic pairing of drivers and riders. More examples can be found in car/bicycle rental business (Oliveira et al. 2018) and parking facility management (Owen and Simchi-Levi 2018). In all the aforementioned applications, customers' willingness for using the service is affected by the price being offered; however, the decision-maker may not know the random demand distributions and how demand reacts to price changes at the beginning of the planning horizon (see, e.g., Xu and Li 2013). Moreover, the revenue does not only depend on the number of customers but also on the service duration of each customer.

This motivates us to consider a general class of revenue management problems with finite reusable resources under incomplete demand-price and service-time information. In our problem, the firm dynamically makes adaptive pricing decisions based on real-time information observed during the past operation, to maximize the total cumulative revenue. The customers arrive at the service system according to a Poisson process, and they request a unit of the resource for an exponentially distributed consecutive time on a first-arrive-first-serve basis. If the resource is available at the time of arrival, the new customer is immediately admitted into the system, and the service immediately starts without delay. If there is no available resource when a customer arrives, she can join the queue and wait. Upon completion of service, the corresponding unit of resource is released and can be directly used to satisfy a new customer. The objective is to learn the mappings between rates and prices from past sales and offer the price dynamically to maximize the total expected revenue.

We propose new multi-armed bandit (MAB) based algorithms for solving this problem. Compared with prior network revenue management and multi-armed bandits literature, the salient difficulties of this problem lie in (i) the unknown rate-and-price mapping information, (ii) the dynamic nature of reusable resources being committed over time, (iii) the transient behavior of the service system when the price changes, and (iv) unbounded and heavy-tailed sub-exponential distributions of observed random variables.

1.2 Dissertation Overview

The remainder of the dissertation contains four main chapters.

In Chapter 2, we consider Benders decomposition for solving two-stage stochastic programs with complete recourse based on finite samples of the uncertain parameters. We propose a learning-enhanced Benders decomposition (LearnBD) algorithm, which adds a cut classification step in each iteration to selectively generate cuts that are more likely to be valuable cuts. We run the LearnBD algorithm on instances of capacitated facility location and multi-commodity network design under uncertain demand. Our results show that SVM cut classifier works effectively for identifying valuable cuts, and the LearnBD algorithm reduces the total solving time of all instances for different problems with various sizes and complexities. The work in Chapter 2 has been published in Jia and Shen (2021).

In Chapter 3, we consider real-time origin-destination transportation requests in, e.g., ride-hailing, and for each request, provide a context-aware route recommendation. To overcome the difficulty of estimating uncertain user preferences toward multiple route features, we design and implement an approach via a combination of the weighted shortest path problem and deep learning, and evaluate it using real-world transportation data. Specifically, the proposed approach learns weights from historical choices of drivers through a deep neural network by minimizing the total weighted costs of historical routes and maximizing those of the non-chosen routes. We evaluate our method and two benchmarks with 4 million requests received by Didi Chuxing in Beijing. Based on the results, we demonstrate that distinguishing request scenarios helps provide preferable context-aware route recommendations. The work in Chapter 3 has been published in Jia et al. (2020a).

In Chapter 4, we consider a price-based revenue management problem with a single type of product consuming a single type of finite reusable resources. The arrival, service, balking and reneging rates depend on the price but the mappings are unknown to the firm. The firm decides prices adaptively to maximize the cumulative revenue. Compared with prior pricing and multi-armed bandit literature, the salient difficulties are (i) unknown rate-and-price mappings, (ii) the dynamic nature of reusable resources being committed over time, (iii) the transient behavior of the system when price changes, and (iv) unbounded and heavy-tailed observed random variables. When the set of prices is finite, we propose two algorithms and prove that the cumulative regrets are both $O(\sqrt{PT\log(T)})$, where T is the number of time periods and P is the cardinality of the price set. When the set of prices is continuous with linear dependence on a d_f-dimensional feature vector, we propose an algorithm and prove a $O(d_f \log(T) \log\log(T)\sqrt{T})$ regret. Both regret results match the lower bound up to a logarithmic factor. As an intermediate step, we also develop a coupling analysis for analyzing the time for a queue to reach the steady state. Our

numerical experiments demonstrate and confirm the efficacy of the proposed algorithms. The work in Chapter 4 has been published online in Jia et al. (2020b).

In Chapter 5, we consider network revenue management problem with multiple products and multiple reusable resources over a finite time horizon. Stochastically arrived customers request a product (service) that occupies a sequence of resources (servers). We also assume that the firm does not know the mappings between customer demand across products and the offered prices, and thus it makes adaptive pricing decisions based only on past sales. Besides all the difficulties in Chapter 4, this problem is more complicated because of the unbalanced utilization ratios among resources and the dynamic nature of service systems with multiple servers being committed and released over time. We propose two new batch MAB type algorihtms for this network revenue management problem and prove that both the proposed algorithms can achieve a cumulative regret $O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\log T\log(|\mathcal{J}|^{1/4}T)}\right)$, where $|\mathcal{J}|$ is the number of products, $|\mathcal{X}|$ is the cardinality of the feasible price set, and T is the total number of time periods. This regret result matches the lower bound up to a logarithmic factor. As part of our regret analysis, we analyze the mixing time for a network queueing system to reach the steady state (from an arbitrary initial state).

CHAPTER 2

Benders Cut Classification via Support Vector Machines for Solving Two-stage Stochastic Programs

We consider Benders decomposition for solving two-stage stochastic programs with complete recourse based on finite samples of the uncertain parameters. We define the Benders cuts binding at the final optimal solution or the ones significantly improving bounds over iterations as *valuable cuts*. We propose a learning-enhanced Benders decomposition (LearnBD) algorithm, which adds a cut classification step in each iteration to selectively generate cuts that are more likely to be valuable cuts. The LearnBD algorithm includes two phases: (i) sampling cuts and collecting information from training problems and (ii) solving testing problems with a support vector machines (SVM) cut classifier. We run the LearnBD algorithm on instances of capacitated facility location and multi-commodity network design under uncertain demand. Our results show that SVM cut classifier works effectively for identifying valuable cuts, and the LearnBD algorithm reduces the total solving time of all instances for different problems with various sizes and complexities.

2.1 Introductory Remarks

In this paper, we focus on the Benders decomposition (Benders 1962) and its implementation for solving a broad class of two-stage stochastic programming models. In the first stage, the value of decision variable $x \in \mathbb{R}^{n_1}$ or $x \in \mathbb{Z}^{n_1}$ is chosen from a feasible region $\mathcal{X}$ before the realization of the uncertainty given the cost vector $c \in \mathbb{R}^{n_1}$. In the second stage, decision variable $y \in \mathbb{R}^{n_2}$ is a continuous recourse decision. The matrix $W \in \mathbb{R}^{m_2 \times n_2}$, vector $h \in \mathbb{R}^{m_2}$, matrix $T \in \mathbb{R}^{m_2 \times n_1}$ and cost vector $q \in \mathbb{R}^{n_2}$ are subject to uncertainty. We denote the overall uncertain parameter as $\xi = [W, h, T, q]$. A two-stage stochastic programming model is given by

$$\min_{x \in \mathcal{X}} \quad c^T x + \mathbb{E}_\xi[Q(x, \xi)] \tag{2.1}$$

where $\mathbb{E}_\xi[\cdot]$ takes expectation of $\cdot$ based on the probability distribution of ξ and

$$\begin{aligned} Q(x,\xi) \stackrel{\text{def}}{=} \min_{y} \quad & q^T y \\ \text{s.t.} \quad & Wy = h - Tx. \end{aligned} \tag{2.2}$$

We consider a finite number of realizations of the uncertain parameter ξ, called "scenarios" or "samples" in the stochastic programming literature (see, e.g., Birge and Louveaux 2011). Let Ω be the sample space that contains all the scenarios and each scenario $\omega \in \Omega$ is associated with a specific realization $\xi_\omega = [W_\omega, h_\omega, T_\omega, q_\omega]$ of the uncertain parameter ξ. Denote the occurrence probability of scenario ω as p_ω, and thus $\sum_{\omega\in\Omega} p_\omega = 1$. Model (2.1) can be reformulated as

$$\min_{x\in\mathcal{X}} \quad c^T x + \sum_{\omega\in\Omega} p_\omega Q_\omega(x), \tag{2.3}$$

where

$$\begin{aligned} Q_\omega(x) \stackrel{\text{def}}{=} Q(x,\xi_\omega) = \min_{y} \quad & q_\omega^T y \\ \text{s.t.} \quad & W_\omega y = h_\omega - T_\omega x. \end{aligned} \tag{2.4}$$

Note that the assumption of having finite scenarios is made without loss of generality. If the uncertain parameter $\xi = [W, h, T, q]$ follows a continuous distribution, one can apply the Monte Carlo sampling approach to generate N_s i.i.d. samples $\{\omega_1, \ldots, \omega_{N_s}\}$ of the uncertain parameters. The second-stage objective function $\mathbb{E}_\xi[Q(x,\xi)]$ can be replaced by the sample average approximation (SAA) $\frac{1}{N_s}\sum_{i=1}^{N_s} Q_{\omega_i}(x)$ (Kleywegt et al. 2002).

2.1.1 An Overview of Benders Decomposition

With large $|\Omega|$, Model (2.3) is in general computationally intractable when it involves integer variable x. The Benders decomposition, which takes advantage of the decomposable structure of two-stage stochastic programs, is applied widely to optimize variants of Model (2.3) formulated for a wide range of applications (see, e.g., Magnanti and Wong 1981). Creating new variables $\theta_\omega \in \mathbb{R}$, $\forall \omega \in \Omega$ in the first-stage problem, one can formulate a relaxation of the original problem, called relaxed master problem (RMP), which has an initial form:

$$(\text{RMP}^0) \quad \min_{x\in\mathcal{X},\theta} \quad c^T x + \sum_{\omega\in\Omega} p_\omega \theta_\omega. \tag{2.5}$$

Subproblems (SPs) are defined as the linear programming dual of the second-stage problems (2.4) with dual variable $\pi_\omega \in \mathbb{R}^{m_2},\ \forall \omega \in \Omega$. We refer to SP$_\omega$ as the SP for scenario ω, formulated as

$$\begin{aligned} (\text{SP}_\omega) \quad Q_\omega^D(x) \ &= \max_{\pi_\omega} \ (h_\omega - T_\omega x)^T \pi_\omega \\ &\text{s.t.} \quad W_\omega^T \pi_\omega \le q_\omega. \end{aligned} \tag{2.6}$$

Let $V^{\omega,t}$ be the set of identified extreme points of the feasible region of SP$_\omega$ in iteration t, and we have $V^{\omega,0} = \emptyset$. Similarly, let $R^{\omega,t}$ be the set of identified extreme rays of the feasible region of SP$_\omega$ in iteration t, and $R^{\omega,0} = \emptyset$. The two sets are respectively associated with Benders optimality cuts and feasibility cuts generated during iterations $1, \ldots, t-1$, and we will explain the cuts in detail later. In iteration t, the corresponding RMP is given by:

$$\begin{aligned} (\text{RMP}^t) \quad \min_{x \in \mathcal{X},\ \theta} \quad & c^T x + \sum_{\omega \in \Omega} p_\omega \theta_\omega \\ \text{s.t.} \quad & \theta_\omega \ge (h_\omega - T_\omega x)^T \nu_\omega \qquad \omega \in \Omega,\ \nu_\omega \in V^{\omega,t}; \\ & (h_\omega - T_\omega x)^T \rho_\omega \le 0 \qquad \omega \in \Omega,\ \rho_\omega \in R^{\omega,t}. \end{aligned} \tag{2.7}$$

After solving RMP^t, we obtain an optimal solution $(\hat{x}^t, \hat{\theta}^t)$. Then for each scenario $\omega \in \Omega$ and its subproblem SP$_\omega$, we first check whether the current RMP^t solution leads to a feasible second-stage problem by solving a corresponding subproblem with decision variable $\sigma_\omega \in \mathbb{R}^{m_2}$, modeled as

$$\begin{aligned} (\text{SP}_\omega\text{-F}) \quad \max_{\sigma_\omega} \quad & (h_\omega - T_\omega \hat{x}^t)^T \sigma_\omega \\ \text{s.t.} \quad & W_\omega^T \sigma_\omega \le \mathbf{0}; \\ & \|\sigma_\omega\| \le 1. \end{aligned} \tag{2.8}$$

If SP$_\omega$-F has a positive optimal objective value with an optimal solution $\bar{\sigma}_\omega$, then from any feasible solution to SP$_\omega$, we can move along the direction $\bar{\sigma}_\omega$ to stay feasible (due to the first constraint of model (2.8)) but increase the objective value of SP$_\omega$. This implies that SP$_\omega$ is unbounded and the second-stage problem is infeasible for given $\hat{x}^t$. To cut off the infeasible first-stage solution $\hat{x}^t$, we generate a Benders feasibility cut:

$$(h_\omega - T_\omega x)^T \bar{\sigma}_\omega \le 0 \tag{2.9}$$

to RMP^{t+1}, which is equivalent to letting $R^{\omega,t+1} = R^{\omega,t} \cup \{\bar{\sigma}_\omega\}$. In this paper, we only focus on the case having complete recourse, under which any feasible first-stage solution will result in a feasible second-stage problem. Therefore, our RMP^t (i.e., Model (2.7)) for each iteration t only contains the first set of optimality cuts, whose derivation is given as follows. If the subproblem

is feasible, then in iteration t we check the optimality of $(\hat{x}^t, \hat{\theta}^t)$. We solve SP$_\omega$ with $x = \hat{x}^t$ to obtain an optimal solution $\bar{\pi}_\omega$ and the optimal objective value $Q^D_\omega(x) = \bar{\pi}^T_\omega(h_\omega - T_\omega \hat{x}^t)$. By strong duality, for any value of x, $Q^D_\omega(x) = Q_\omega(x)$. The solution to RMP^t will reach the same objective value as the original problem when $\hat{\theta}^t_\omega \geq Q_\omega(\hat{x}^t),\ \forall \omega \in \Omega$. Therefore, $\hat{\theta}^t_\omega < Q^D_\omega(x)$ indicates that the current solution $(\hat{x}^t, \hat{\theta}^t)$ is not optimal for the original problem. Thus, we add a Benders optimality cut

$$\theta_\omega \geq (\bar{\pi}_\omega)^T(h_\omega - T_\omega x) \tag{2.10}$$

to RMP^{t+1}, which is equivalent to letting $V^{\omega,t+1} = V^{\omega,t} \cup \{\bar{\pi}_\omega\}$. We refer to the cuts being added and the corresponding dual extreme points being identified interchangeably in this paper.

In iteration t, the objective value of RMP^t provides a valid lower bound to the origin problem because it is a relaxation. If all SPs have finite optimal objective values, $\hat{x}^t$ and all recourse solutions together form a feasible solution to the original two-stage problem, and thus

$$c^T\hat{x}^t + \sum_{\omega \in \Omega} p_\omega Q_\omega(\hat{x}^t) \tag{2.11}$$

provides a valid upper bound. The algorithm terminates when the upper and lower bounds are equal or their gap is within a pre-specified tolerance δ. In this paper, we define the gap as

$$\text{optimality gap} = \frac{\text{upper bound} - \text{lower bound}}{\text{lower bound}} \times 100\%. \tag{2.12}$$

The Benders decomposition converges in a finite number of iterations due to the finite number of dual extreme rays and extreme points of the finitely many SPs.

2.1.2 Challenges and Research Overview

While the Benders decomposition method helps to solve two-stage stochastic programs efficiently, it could suffer from slow convergence. One reason is that the size of RMPs becomes too large due to the quickly increased number of newly added cuts over iterations. Geoffrion and Graves (1974) are among the first to notice and emphasize on the computational difficulty of solving RMPs for stochastic binary integer programs. Magnanti and Wong (1981) report that over 90% of the total time of implementing the Benders decomposition is spent on solving RMPs. Minoux (1986) points out that not all extreme points of the feasible region of SPs equally contribute to restricting the optimal solution to RMPs. Therefore, a larger number of Benders cuts are not tight at the final optimal solution, but can increase the size of RMPs, which are then extremely hard to solve as large-scale integer programs.

We propose a two-phase learning-enhanced Benders decomposition (LearnBD) algorithm to

solve two-stage stochastic integer programs with finite samples of the uncertain parameter and complete recourse, where the second-stage subproblems are linear programs (LPs). We define cuts as *valuable cuts* when they can either cut the feasible region in the current iteration significantly, or be tight at the final optimal solution (see Holmberg 1990, for a similar definition in the latter case). Our goal is to only add valuable cuts to the corresponding RMP in each iteration. Up to date, there is no practical and systematic way to perform cut classification and to accelerate the iterative process for Benders decomposition for large-scale optimization problems, according to Rahmaniani et al. (2017). We propose to integrate machine learning techniques into the traditional Benders decomposition framework to learn cut characteristics and selectively generate subsets of Benders cuts iteratively.

2.1.3 Contributions of the Paper

We summarize the main contributions of this paper as follows.

- Firstly, we identify a set of characteristics and quantify performance measures of Benders cuts. We construct a cut classifier using support vector machines (SVM), a widely used supervised machine learning method that takes history observations and their labels as input, to identify valuable cuts in each iteration.

- Secondly, we develop the LearnBD algorithm with SVM cut classifier, to limit the size of RMPs and reduce total solving time. We also provide guidelines for choosing hyperparameters for enhancing the effectiveness of the LearnBD algorithm.

- Thirdly, we test instances of capacitated facility location and multi-commodity network design under uncertain demand, to demonstrate the computational advantages of LearnBD in different problem settings. Our results show that the LearnBD algorithm leads to smaller sizes of RMPs with fewer accumulated cuts, and therefore shorter time for solving RMPs.

2.1.4 Structure of the Paper

The remainder of this paper is organized as follows. In Section 2.2, we review the literature on the effort of improving Benders decomposition for solving large-scale optimization problems. In Section 2.3, we develop the LearnBD algorithm and use SVM for constructing the cut classifier. In Section 2.4, we present the computational results of the LearnBD algorithm benchmarked with the traditional Benders approach. In Section 2.5, we conclude the paper and describe future research.

2.2 Literature Review

The Benders decomposition was initially proposed by Benders (1962) and was then widely used for solving problems of scheduling and planning (Cordeau et al. 2001, Hooker 2007), network optimization and transportation (Laporte et al. 1994, Costa 2005, Binato et al. 2001), and inventory control and management (Federgruen and Zipkin 1984, Cai et al. 2001). Magnanti and Wong (1981) and Naoum-Sawaya and Elhedhli (2013) note that directly applying the traditional Benders decomposition may require excessive computational effort. It is mainly due to the poor convergence of RMPs that has been computationally demonstrated in Orchard-Hays et al. (1968) and Wolfe (1970). Several researchers have proposed enhancement strategies depending on different problem structures to accelerate the algorithm accordingly, of which we describe the details below.

In the traditional Benders approach detailed in Section 2.1.1, RMPs and SPs are solved iteratively, and thus the first stream of studies concentrates on problem-solving techniques, and particularly techniques for efficiently computing RMPs or SPs. Geoffrion and Graves (1974) propose to only sub-optimally solve RMPs in each iteration to enable cut generation, without seeking tight cuts at the beginning of the Benders approach. Similarly, Raidl (2015) solves RMPs using heuristics to save computational time. Zakeri et al. (2000) show that sub-optimal solutions to the SPs can still generate valid cuts in RMPs, and thus effective heuristic approaches are designed for solving the SPs approximately.

The second stream of studies focuses on decomposition strategies, to guide the process of partitioning variables to remain in RMPs or in SPs. Crainic et al. (2014) propose a partial Benders decomposition algorithm to reduce the number of feasibility and optimality cuts, while adding information of SPs into RMPs by retaining or creating scenarios. They develop different decomposition strategies for choosing the retaining scenarios when solving two-stage mixed-integer programming (MIP) models with continuous recourse. In addition, Gendron et al. (2016) propose a non-standard decomposition strategy, which retains the second-stage variables in RMPs, and the authors test the results using instances of network design problems.

Machine learning techniques have been applied to general-purpose optimization algorithms for urging quick convergence to optimal or sub-optimal solutions (see, e.g., He et al. 2014b, Khalil et al. 2017). Khalil et al. (2016) introduce a novel data-driven framework for variable selection to solve MIP models via branch-and-bound algorithm efficiently. Kruber et al. (2017) develop a supervised learning approach to distinguish a stronger reformulation of a given MIP model and to determine which decomposition to implement in order to improve the speed of MIP solvers. Misra et al. (2018) directly construct a model for seeking the optimal solution as a function of the input parameter, by learning relevant sets of active constraints given computationally expensive large-scale parametric models.

Recently, several papers apply machine learning to improve algorithmic efficiency of decomposition approaches, especially focusing on cut classification. Among them, Tang et al. (2019) model the cut selection in integer programming as a reinforcement learning (RL) problem. They define the corresponding concepts in RL and implement an offline training phase. Baltean-Lugojan et al. (2019) develop linear outer-approximations of semi-definite constraints that can be effectively integrated into global solvers. They construct a neural network for predicting the objective improvement of each cut, which is similar to the performance measure proposed in our paper.

2.3 Learning-enhanced Benders Decomposition

We develop a Learning-enhanced Benders Decomposition (LearnBD) for solving two-stage stochastic programs with complete and continuous recourse in the second stage. The algorithm aims to solve a set of two-stage problems that share similar problem structures (i.e., dimensions of decision variables, constraint matrices, and cost parameters in the objective function) but could have different realizations of the uncertain parameter. As a result, LearnBD constructs a training problem that shares the same problem structures as the original problem(s), while the distributions of the uncertain parameter can be different. LearnBD samples cut and collects information from the training problem. Then, it uses the collected cut information to train an SVM cut classifier and then optimizes the original problem(s).

To find potential applications of LearnBD, consider some industries where we need to periodically solve similar optimization problems with the same system structures and decision frameworks, but with different input data representing the current environment and status of the system. For example, the unit commitment problem in the power system is solved every hour to determine the operational schedule of the generating units under random renewable generation and electricity loads (see, e.g., Saravanan et al. 2013, Dashti et al. 2016). A grid operator needs to solve a stochastic program in the form of (2.3) every hour with different ξ_ω-values. If one can solve these similar problems in an efficient way, it can significantly improve the operational efficiency of power grids. The proposed LearnBD can be applied in this case, where one can sample cuts from training problems using previous days' data , and re-use the cut classifier for the stochastic programs to be solved in future hours. Moreover, even for the problems that we only solve once, LearnBD could be useful. For instance, consider solving stochastic programming models using SAA, where we can solve a number of SAA-based reformulations with different i.i.d. samples repeatedly, by using cut information collected from solving one such reformulation as a training problem.

LearnBD includes two phases: Offline cut sampling (Phase 1) and solving a given problem using cut classification (Phase 2). The collected cut information can be used to solve any testing problem of the same variable-and-constraint size. Therefore, the time spent on Phase 1 does not

affect the total solving time. In Phase 1, we solve training problems under the estimation of the uncertainty and collect training data for cut classification. In Phase 2, for a given testing problem, which is viewed as an unseen testing instance, we train cut classifiers using the training data from Phase 1 and apply cut classification steps throughout the Benders iterations.

We provide an overview of LearnBD in Figure 2.1, in which related to Phase 1, K is the number of sampling paths, N is the length of each sampling path, and RMP_k^n, $n = 1, \ldots, N$, $k = 1, \ldots, K$ is the RMP of the training problem corresponding to iteration n in sampling path k. Related to Phase 2, RMP^t, $t = 0, \ldots, T$ is the RMP of the testing problem corresponding to iteration t and $t = T$ denotes the last iteration. In Phase 1, we perform cut sampling to generate training data, including cut characteristics and performance measures (see Section 2.3.1). In Phase 2, we utilize the classifier to distinguish valuable cuts from all generated cuts in each iteration and solve RMPs iteratively by only adding valuable cuts (see Section 2.3.3). As a sub-procedure in Phase 2, we train an SVM classifier with the training data generated in Phase 1, which takes cut characteristics as input and $\{1, -1\}$ valued label as output to classify whether or not a cut is valuable (see Section 2.3.2).

2.3.1 Phase 1: Cut Sampling

In Phase 1, we conduct cut sampling from some training problem to collect the information of valuable cuts, which will then be used to train the classifier in Phase 2. The training data set D can be viewed as a $D_{row} \times D_{col}$ matrix, where each row is the information of a specific sample cut, the first $D_{col} - 1$ columns are cut characteristics, and the last column is a $\{-1, 1\}$ valued label.

Characteristics. Cut characteristics are features of a cut that can help us predict the performance of the cut in future iterations if it is added to the current RMP. We consider the following two characteristics. The first is cut violation at the current solution $(\hat{x}^t, \hat{\theta}^t)$ of RMP^t, denoted by VL and it can be computed as $\pi_\omega^T(h_\omega - T_\omega \hat{x}^t) - \hat{\theta}_\omega^t$, according to (2.10). This characteristic reflects how large the feasible region of RMP^t can be cut off if adding the cut. The second characteristic is related to the scenario where a cut is generated from. We denote the number of cuts generated by the same scenario in previous iterations as NC. This characteristic reflects the trade-off between exploration and exploitation, two typical learning strategies. A preference to a cut whose associated scenario generates more cuts in previous iterations, links to an exploitation strategy, while the opposite preference leads to an exploration strategy. On the one hand, a large number of cuts generated from the same scenario shows that this scenario is crucial for identifying an optimal solution. However, it could be the case where the majority of valuable cuts in this scenario have already been generated. Thus, the change of the objective value of RMP brought by a new cut from the

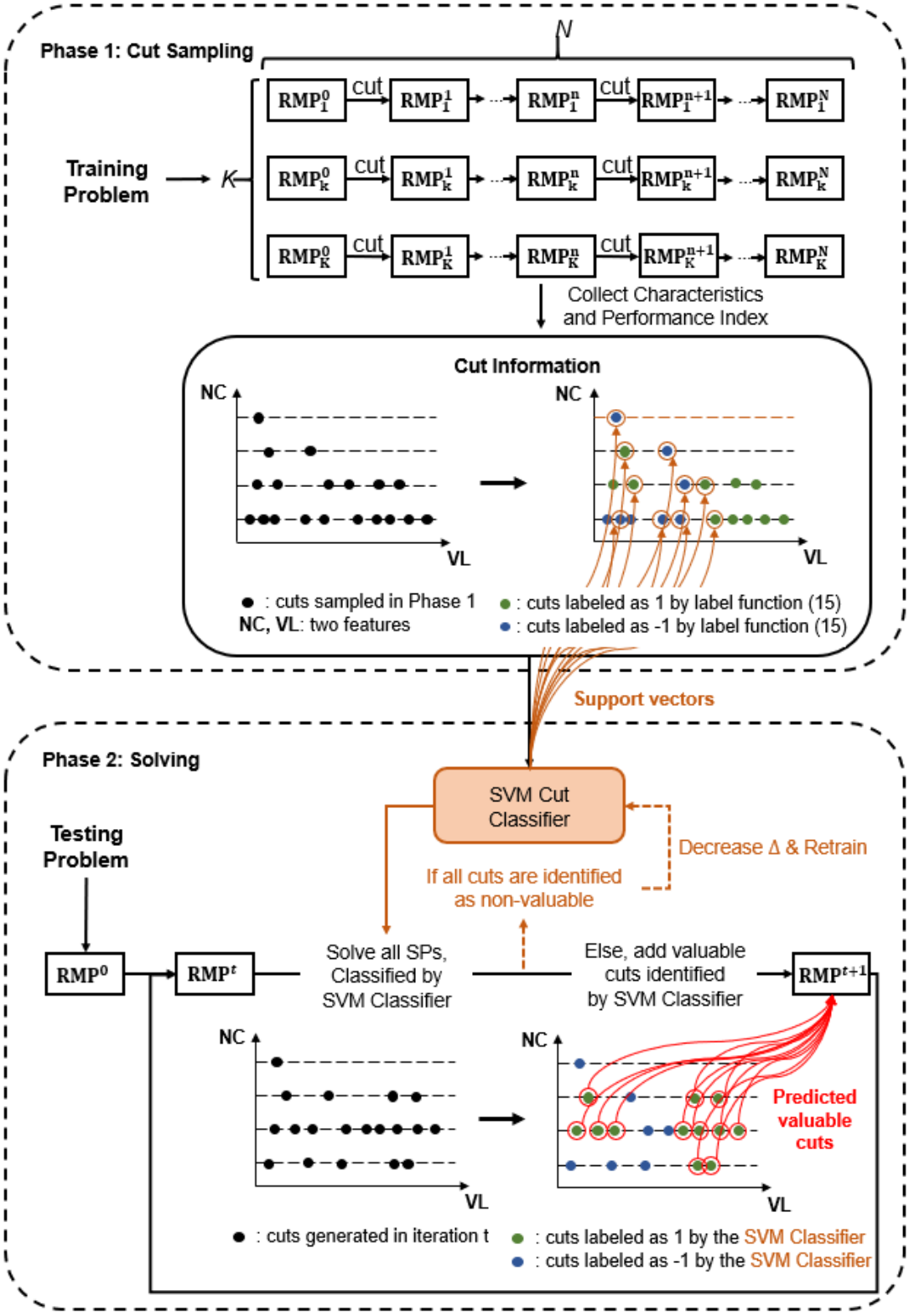


Figure 2.1: An overview of LearnBD procedures. In Phase 1, we collect cuts from training problems and label them as valuable cuts (green points) and non-valuable cuts (blue points). Then, we train an SVM classifier and use it to classify the new cuts we meet in Phase 2 when solving the testing problem. In Phase 2, we only include the valuable cuts identified by the SVM classifier in each iteration.

same scenario can be small. Therefore, the relationship between NC and future performance of a cut is highly possible to be nonlinear. A collection of characteristics of one specific cut is referred to as an observation $\mathbf{o}$, with $\mathbf{o} = (\text{VL}, \text{NC})$.

Performance index. In the training data set, each observation also needs to be assigned a label l, where 1 is assigned to valuable cuts and -1 is assigned to non-valuable cuts. Therefore, we define a performance index of each cut and then transform it into $\{-1, 1\}$ valued label. We choose the change amount of the objective value of RMP^t before and after adding a cut as the performance index of the cut, denoted by PI. We add exactly one cut to RMP^t each time to recognize the change of objective value brought by the cut. In practice, users can customize the characteristics and performance index according to specific applications. The rule for transforming the performance index will be discussed after we introduce sampling paths next.

Sampling path. We construct sampling paths to guide the cut sampling process to record the cut characteristics and performance index. The number of sampling paths K and the length of sampling path N are pre-determined hyperparameters. In each sampling path $k,\ k = 1, \ldots, K$, we start with RMP^0_k, which is initialized by RMP^0. In iteration n of a sampling path k, for $n = 0, \ldots, N-1,\ k = 1, \ldots, K$, we solve RMP$^n_k$ and obtain an optimal solution $(\hat{x}^n_k, \hat{\theta}^n_k)$. Then we follow the Monte Carlo sampling approach to randomly sample one scenario $\omega \in \Omega$ and solve the corresponding SP$_\omega$ by plugging in $(\hat{x}^n_k, \hat{\theta}^n_k)$: (i) if no optimality cut is generated, then we continue sampling another scenario and solving the corresponding SP; (ii) if an optimality cut is generated, we record the two characteristics, instantly add the cut to RMP$^{n+1}_k$, and then record the performance index. Similar to Section 2.1.1, we use $V^{\omega,n}_k$ to denote the set of identified extreme points of the feasible region of SP$_\omega$ in iteration n of sampling path k, and RMP^n_k is defined in the following form:

$$
\begin{aligned}
(\text{RMP}^n_k) \quad & \min_{x \in \mathcal{X},\, \theta} \ c^T x + \sum_{\omega \in \Omega} p_\omega \theta_\omega \\
& \text{s.t.} \quad \theta_\omega \geq (h_\omega - T_\omega x)^T \nu_\omega \qquad \omega \in \Omega,\ \nu_\omega \in V^{\omega,n}_k.
\end{aligned}
\tag{2.13}
$$

Once a new cut is generated, we move one step forward in one sampling path and therefore the iterative process stops after reaching RMP^N_k in sampling path $k,\ k = 1, \ldots, K$.

Through cut sampling, we collect $\Gamma = N \times K$ number of training data. A larger set of training data in general leads to a more precise classifier. Larger N means that we can collect information of more representative cuts because we solve RMPs in a wider range of problem sizes. Different independent sampling paths can be conducted in parallel, and therefore, larger K will not significantly increase the time of Phase 1. However, the cuts generated by RMPs with similar sizes can share similar characteristics. These similar inputs can also lead to over-fitting and can eventually

Algorithm 2.1 Phase 1 of the LearnBD algorithm.

1: **Input:** a two-stage stochastic program with a set Ω of scenarios; values of N, K, Δ.
2: **Initialize:** RMP^0 and SP_ω, $\forall \omega \in \Omega$ of the training problem, $D \leftarrow \emptyset$.
3: **for** $k = 1, \ldots, K$ **do**
4: **Initialize**: $V_k^{\omega,0} \leftarrow \emptyset,\ \forall \omega \in \Omega$, $\text{NC}_\omega = 0,\ \forall \omega \in \Omega$, $\text{RMP}_k^0 \leftarrow \text{RMP}^0$, $D_{\text{temp}} \leftarrow \emptyset$.
5: Solve RMP_k^0, obtain an optimal solution $\{\hat{x}_k^0, \hat{\theta}_k^0\}$ with optimal objective value $\hat{z}_k^0$.
6: **for** $n = 0, \ldots, N-1$ **do**
7: Randomly select $\omega' \in \Omega$, solve $\text{SP}_{\omega'}$, obtain an optimal solution $\pi_{\omega'}$ and its objective value $\zeta_{\omega'}$.
8: **if** $(\hat{\theta}_k^n)_{\omega'} < \zeta_{\omega'}$ **then**
9: $V_k^{\omega,n+1} \leftarrow V_k^{\omega,n} \cup \{\pi_{\omega'}\}$, $\text{NC}_{\omega'} \leftarrow \text{NC}_{\omega'} + 1$, $\text{VL} \leftarrow \zeta_{\omega'} - (\hat{\theta}_k^n)_\omega$;
10: **else**
11: Go to Step 7.
12: **end if**
13: Solve RMP_k^{n+1}, obtain an optimal solution $\{\hat{x}_k^{n+1}, \hat{\theta}_k^{n+1}\}$ with optimal objective value $\hat{z}_k^{n+1}$, $\text{PI} \leftarrow |\hat{z}_k^{n+1} - \hat{z}_k^n|$,
14: $(D_{\text{temp}})_{n+1,\cdot} \leftarrow (\text{VL}, \text{NC}_{\omega'}, \text{PI})$.
15: **end for**
16: $D \leftarrow D \cup ((D_{\text{temp}})_{N,1}, (D_{\text{temp}})_{N,2}, 1)$
17: **for** $n = N-1, \ldots, 1$ **do**
18: **if** $\frac{(D_{\text{temp}})_{n,3}}{(D_{\text{temp}})_{n+1,3}} < \Delta$ **then**
19: $D \leftarrow D \cup ((D_{\text{temp}})_{n,1}, (D_{\text{temp}})_{n,2}, -1)$
20: **else**
21: $D \leftarrow D \cup ((D_{\text{temp}})_{n,1}, (D_{\text{temp}})_{n,2}, 1)$
22: **end if**
23: **end for**
24: **end for**
25: **return** D

weaken the power of the classifier. In our later computational studies, we choose $N = 2 \times |\Omega|$ and $K = 2$.

Remark 2.3.1. *The cut sampling process is independent across all sample paths, and thus the cut information collected in different sampling paths is independent of one another.*

Remark 2.3.2. *These sampled cut information can be re-used in Phase 2 when solving different testing problems and thus the time of Phase 1 does not affect the total solving time of testing problems.*

2.3.2 Subroutine in Phase 2: Classifier Construction

We introduce a subroutine in Phase 2, i.e., constructing SVM classifiers with training data D, to predict the potential performance of cuts and identify valuable cuts. As mentioned in Sec-

tion 2.3.1, D can be presented as a collection of observations and labels of sample cuts, where $D = \{(\mathbf{o}_d, l_d),\ d = 1, \ldots, \Gamma\}$ and more specifically each cut observation $\mathbf{o} = (\text{VL}, \text{NC})$. In Section 2.3.2.1, we show how SVM works and how to estimate the parameters with training data. In Section 2.3.2.2, we discuss the advantages of SVM as a cut classifier.

2.3.2.1 Building Cut Classifier Using Support Vector Machines (SVM).

SVM is a well-known supervised machine learning approach (see Cortes and Vapnik 1995, Vapnik 1998, 1999, 2013) and has been used for analyzing data in many applications. Given a training data set $D = \{(\mathbf{o}_d, l_d),\ d = 1, \ldots, \Gamma\}$ from Phase 1, where $\mathbf{o}_d \in \mathbb{R}^{\Sigma}$ is an observation (in our problem $\Sigma = 2$), a subset of training data are identified as support vectors after the training process. Parameterized by a coefficient vector $\mathbf{a} \in \mathbb{R}^{\Gamma}$, and an intercept $b \in \mathbb{R}$, the SVM classifier $f_{SVM}(\cdot) : \mathbb{R}^{\Sigma} \to \{-1, 1\}$ for a new observation $\mathbf{o}' \in \mathbb{R}^{\Sigma}$ (i.e., the collected information of a specific cut in our problem), is given by

$$f_{SVM}(\mathbf{o}') = \text{sign}\left[\sum_{d=1}^{\Gamma} l_d a_d K(\mathbf{o}', \mathbf{o}_d) + b\right], \tag{2.14}$$

where $K(\cdot, \cdot) :\ \mathbb{R}^{\Sigma} \times \mathbb{R}^{\Sigma} \to \mathbb{R}$ is a predetermined kernel function. Here we use one of the most popular kernel functions, the Radial Basis Function (RBF), in which $K(\mathbf{o}_1, \mathbf{o}_2) = \exp(-\gamma \|\mathbf{o}_1 \cdot \mathbf{o}_2\|^2)$.

Label transformation. We define a label transformation function to transform continuous performance index into $\{-1, 1\}$ label, where 1 indicates a valuable cut. The nature of the convergence of Benders decomposition is accompanied by the fact that the change of the objective value of RMPs is decreasing over iterations. Therefore, we treat the cuts that can bring a large enough proportion of PI of the next cut in the same sampling path as valuable cuts. We directly assign label 1 to the last cut of each cut sampling path. For other cuts, we calculate the ratio of its PI and the PI of the next cut in the same sampling path and then compare the ratio with a pre-determined threshold $\Delta \in [0, 2]$. The label transformation function is defined as:

$$l_k^n = \begin{cases} -1, & \text{if } \dfrac{\text{PI}_k^n}{\text{PI}_k^{n+1}} < \Delta \\ 1, & \text{otherwise,} \end{cases} \quad n = 0, \ldots, N-1,\ k = 1, \ldots, K \tag{2.15}$$

Larger Δ shows a more strict rule for recognizing a cut as a valuable cut. With this label transformation function, one can calculate all labels using current performance indices and use these labels to train an SVM classifier. We present the algorithmic details in Algorithm 2.1.

Remark 2.3.3. *Label transformation function eliminates a degree of dependency across cuts generated in the same sampling path. Together with Remark 2.3.1, all training data are independent with each other.*

The label prediction function $f_{SVM}(\cdot)$ can be interpreted as follows. We can treat the coefficient a_d as a significance-magnitude of the corresponding data point $d = 1, \ldots, \Gamma$, because the label l_d is always shown in $a_d \times l_d$ in the predicting process and $a_d \times l_d$ as a whole indicates the power of data point d for classifying new cuts. The kernel function $K(\mathbf{o}', \mathbf{o}_d)$ presents the similarity between the characteristics of a new cut $\mathbf{o}'$ and cut $\mathbf{o}_d$. Then the label of $\mathbf{o}'$ is the sign of a sum of magnitude-adjusted label of all training data points plus an intercept b. Then by eliminating the training data with zero estimated coefficients a_d, the remaining training data form the support vector set S, which is a subset of D, and function (2.14) can be simplified as

$$f_{SVM}(\mathbf{o}') = \text{sign}\left[\sum_{\mathbf{s}\in S} l_s a_s K(\mathbf{o}', \mathbf{o}_s) + b\right]. \tag{2.16}$$

The parameters $\{S, \mathbf{a}, b\}$ can be trained by minimizing the prediction loss function among training data as well as maximizing the flatness of the boundary between valuable and non-valuable cuts. The prediction loss is computed by the hinge loss function to improve the model sparsity. Given the estimated result $u = \sum_{d=1}^{\Gamma} l_d a_d K(\mathbf{o}', \mathbf{o}_d) + b$ from Equation (2.14) and the ground truth label l, the loss is calculated by:

$$\text{Loss}(u, l) = \max(0,\ 1 - l \cdot u). \tag{2.17}$$

It can be seen that when l and u have the same sign and $|u| \geq 1$, the loss $= 0$; otherwise the loss $= |u - l|$. For brevity, we elaborate on the training process of SVM in Appendix A.1.

Remark 2.3.4. *The penalty hyperparameter C balances the explanatory and predictive power of the classifier. In general, a larger C shows a smaller tolerance of prediction error within the training dataset and hence results in a classifier with higher explanatory power while too large C will destroy the predictive power. The discount rate γ in RBF kernel determines the magnitude of similarity between observations, which is related to the model sensitivity and convergence property. Proper (γ, C) will generate a relatively small number of support vectors with an accepted classification accuracy. The classification accuracy is defined as the percentage of the given cuts whose predicted labels are the same as the input labels. In our later computational studies, the classification accuracy of classifiers on the training data is almost 100%. Those two hyperparameters are generally selected together via cross-validation and grid search to reach the best empirical performance. Typically, the larger C we use, the more support vectors can be identified by the classifier,*

and thus the classification effort will increase. The classification accuracy on the training data can be improved, while the accuracy on unseen testing data can be impaired.

2.3.2.2 Reasons for Choosing SVM.

The advantage of using support vector type of methods for cut classification is threefold. Firstly, with the help of the hinge loss function, SVM only selects representative observations from the training data. Those observations are referred to as support vectors and are stored for future classification. This sparse nature increases the computational speed for evaluating new cuts. Secondly, the mechanism of SVM can be explained by using the similarity between a new cut and all support vectors to predict future performance, which is consistent with our assumptions and motivation that valuable cuts share similarities. Furthermore, the kernel-based method can flexibly help capture the nonlinear relationship between cut performance and characteristics. Thirdly, the solving process of SVM is a convex optimization problem (see model (SVM-P) in (A.2)) which is computationally tractable.

Another support vector type of learning method is support vector regression (SVR) (see, e.g., Smola 2004). The main idea of those two approaches is similar. SVM classifies cuts by $\{1, -1\}$ labels and works as a classifier. SVR evaluates continuous scores of cuts and works as a regressor, which is more informative than a classifier because it can distinguish more rank levels and also allows any fractional rank between levels. We choose SVM over SVR following concerns listed as follows. Indeed, cuts have different levels of effectiveness for improving RMP solutions. However, we choose not to spend time and effort to fully distinguish between those levels. Recall that the geometric explanation of Benders decomposition is to cut off the feasible region of RMP^t in each iteration t. The cuts generated in one iteration can be linearly independent with each other, and thus they cut the feasible region from different directions. Therefore, it is better to include several valuable cuts rather than only one cut in each iteration. On the other hand, since we do not select the cuts with relatively low effectiveness, we do not even need to distinguish among those non-valuable cuts. A similar reason is also mentioned in the learning approach used for a branch-and-bound algorithm by Khalil et al. (2016).

Tang et al. (2019) employ a neural network for selecting cuts for solving integer programming models. One advantage of utilizing a neural network is the complex and high-dimensional data it can handle and process. The information we use contains coefficients of the current cut and those of all added cuts. If we use a neutral network, it can evaluate each cut adaptively to the solving process. In LearnBD, we achieve this iteration-adaptive property by allowing retraining of the cut classifier (see Remark 2.3.5). The training time of SVM classifier is much shorter than that of neural networks, and the classifier can be trained before solving the testing problems (see Remark 2.3.2 and Remark 2.4.1).

2.3.3 Phase 2: Cut Classification

Iteration rule. In Phase 2, we solve a given two-stage model in the form of (2.3), which is also referred to as the testing problem. In iteration t, we solve RMP^t of the testing problem and obtain an optimal solution $(\hat{x}^t, \hat{\theta}^t)$; by plugging in $(\hat{x}^t, \hat{\theta}^t)$, we solve all SP$_\omega$, $\forall \omega \in \Omega$ of the testing problem and record the two characteristics of each generated cut. Using the characteristic information, the SVM classifier assigns label 1 to valuable cuts and -1 to non-valuable cuts (see Section 2.3.2). Then we add all valuable cuts with label $= 1$ to RMP^{t+1}. We use $\overline{V}^{\omega,t}$ to denote the identified extreme points of the feasible region of SP$_\omega$ in iteration t in Phase 2. We repeat adding cuts until the optimality gap between upper bound and lower bound, which is defined in (2.12), is less than a pre-specified tolerance δ. If no cut is labeled 1 by SVM classifier, but we have not reached the optimal tolerance, then we retrain the cut classifier with a smaller Δ and continue iterating. We define a decreasing list $\mathcal{L}_\Delta$ as potential values of Δ, and retrain the SVM classifier by plugging in $\Delta = \mathcal{L}_\Delta(l)$ in the lth retraining. The algorithmic details of Phase 2 are presented in Algorithm 2.2.

Remark 2.3.5. *As mentioned in Section 2.3.2.2, we allow retraining to achieve the iteration-adaptive property of Benders decomposition. If we define the state of the two-stage optimization problem as the set of added cuts to RMP, then the number of possible states is extremely large because the new cuts are also directly affected by the previous solving trajectory. Thus, it is highly likely that the state we see during the solving process may not have been encountered during the training data collection process and consequently, the relationship between the cut features and valuable labels may not reflect the true relationship. In the context of machine learning, this problem, induced by encountering unseen data points, is also defined as distribution shift. When distribution shift happens, the previous classifier does not work anymore for predicting the labels of cuts generated in an unseen state. Actually, distribution shift may happen in several machine learning algorithms while solving sequential decision-making problems, such as algorithms based on behavioral cloning in Imitation Learning (see, e.g., Pomerleau 1991), and thus several studies focus on remedying distribution shift (see, e.g., Ross et al. 2011, Reddy et al. 2019). In this paper, we propose to mitigate distribution shift by retraining the SVM classifier with decreasing Δ in* (2.15)*, or equivalently, we enforce a less strict standard for valuable cuts in later iterations.*

Remark 2.3.6. *In algorithmic steps, the SVM classifier is retrained several times in Phase 2. In practical implementation, one can train classifiers with different values of Δ before starting Phase 2 and call classifiers with the specific Δ-value when solving a problem. Thus, these classifiers can be re-used and the training time of classifiers does not affect the total solving time of testing problems. We summarize the numerical performance of retraining in Remark 2.4.1 in Section 2.4.3.2.*

Algorithm 2.2 Phase 2 of the LearnBD algorithm.

Input: RMP^0 and $\text{SP}_\omega, \forall \omega = 1, \ldots, \Omega$ of the testing problem, value of δ.
Initialize: set of generated cuts $\overline{V}^{\omega,0} \leftarrow \emptyset$ and number of cuts $\overline{\text{NC}}_\omega \leftarrow \emptyset,\ \forall \omega \in \Omega$, list $\mathcal{L}_\Delta$, $l = 1$, train an SVM classifier with $\Delta = \mathcal{L}_\Delta(l)$.
Solve RMP^0, obtain an optimal solution $\{\hat{x}^0, \hat{\theta}^0\}$ and optimal objective $\hat{z}^0$, $t \leftarrow 0$, UB $\leftarrow +\infty$, LB$\leftarrow \hat{z}^0$.
while $\frac{\text{UB}-\text{LB}}{\text{LB}} > \delta$ **do**
  $n_{cut} \leftarrow 0$.
  for $\omega \in \Omega$ **do**
    Solve SP_ω, obtain an optimal solution π_ω and its optimal objective value ζ_ω.
    if $(\hat{\theta}^t)_\omega < \zeta_\omega$ **then**
      VL $\leftarrow \zeta_\omega - (\hat{\theta}^t)_\omega$.
      Input $\{\text{VL}, \overline{\text{NC}}_\omega\}$ into SVM classifier.
      if Predicted label is 1 **then**
        $\overline{V}^{\omega,t+1} \leftarrow \overline{V}^{\omega,t} \cup \{\pi_\omega\}$; $n_{cut} \leftarrow n_{cut} + 1$; $\overline{\text{NC}}_\omega \leftarrow \overline{\text{NC}}_\omega + 1$.
      end if
    end if
  end for
  if $n_{cut} = 0$ **then**
    $l = l + 1$, train an SVM classifier with $\Delta = \mathcal{L}_\Delta(l)$; Continue.
  end if
  UB $\leftarrow \min\{\hat{z}^t + \sum_{\omega \in \Omega} \zeta_\omega\}$.
  $t \leftarrow t + 1$, re-solve RMP^t, obtain an optimal solution $\{\hat{x}^t, \hat{\theta}^t\}$, optimal objective $\hat{z}^t$.
  LB $\leftarrow \max\{LB, \hat{z}^t\}$.
end while
return Optimal solution $\{\hat{x}^t, \hat{\theta}^t\}$, optimal objective value $\hat{z}^t$.

2.4 Numerical Studies

We evaluate LearnBD on two classes of stochastic programs: (i) Capacitated facility location problem (CFLP) and (ii) fixed charge multi-commodity network design problem (CMND). CFLP contains binary first-stage variables and continuous second-stage variables with complete recourse. CMND contains binary first-stage variables and continuous second-stage variables. The traditional formulation of CMND (see, e.g., Crainic et al. 2014) also involves feasibility cuts in the Benders decomposition. We introduce auxiliary variables in the formulation so that the complete recourse assumption holds. These problems naturally appear in many applications (see, e.g., Melkote and Daskin 2001, Klibi et al. 2010, Klibi and Martel 2012), and they are notoriously hard to solve (see, e.g., Geoffrion and Graves 1974, Birge and Louveaux 2011, Crainic et al. 2001, 2011).

Section 2.4.1 describes our experimental design and computational settings. Section 2.4.2 shows prediction accuracy of the SVM Classifier over different validation sets. Section 2.4.3 presents the overall results of diverse-sized instances and Section 2.4.4 presents detailed computational results over iterations. Section 2.4.5 presents the time spent on Phase 1 and on training SVM classifiers. Section 2.4.6 provides results of LearnBD using a classifier trained with cut information collected from another instance. All the tests are performed on a computer with an Intel Core E5-2630 v4 CPU 2.20 GHz and 128 GB of RAM.

2.4.1 Experimental Setup and Test Instances

2.4.1.1 Capacitated Facility Location Problem (CFLP).

Consider a set W of production plants (facilities) and a set F of factories which have uncertain demand $\tilde{d}$. The setup cost of facility i, $\forall i \in W$ is k_i and the production capacity limit is u_i. The demand of factory $j, \forall j \in F$ is uncertain and can be satisfied by products produced in facility i, $\forall i \in W$ if it is open with a unit transportation cost c_{ij}, and the unmet demand will generate lost-sale with a unit penalty cost ρ_j. One needs to decide a subset of facilities to open before the realization of the demand to minimize the expected total cost. We provide the details about RMP and SP formulations in Appendix A.2.1.

For our studies, we use problem sets IV and VI in Beasley (1988), which are originally from Akinc and Khumawala (1977) and Christofides and Beasley (1983). Each problem set uses the same network and identical capacity among facilities. Table 2.1 summarizes the attributes of the instances, which are originally proposed for the deterministic capacitated facility location problem, and we apply the techniques in Song et al. (2014) for sampling scenarios. The demand $\tilde{d}_j$ of factory j in each scenario ω follows a Normal distribution with mean equal to the demand used in the original deterministic instances and standard deviation equal to $0.1 - 0.2$ times the mean.

Table 2.1: Instance Attributes of CFLP

Problem Set	$\|W\|$	$\|F\|$	Capacity u	Setup Cost k
IV	16	50	5000	7.5 / 12.5 / 17.5 / 25
VI	16	50	15000	12.5

2.4.1.2 Multi-commodity Network Design Problem (CMND).

Consider a directed network with node set N, arc set A, and commodity set K. An uncertain $\tilde{v}_k$ amount of commodity $k, \forall k \in K$ must be routed from an origin node, $o_k \in N$, to a destination node, $d_k \in N$. The installation cost and arc capacity of arc (i, j), $\forall (i, j) \in A$ are f_{ij} and u_{ij}, respectively. The cost for transporting one unit of commodity k, $\forall k \in K$ on installed arc (i, j), $\forall (i, j) \in A$ is c^k_{ij}. One needs to decide a subset of arcs to install before the realization of the demand to minimize the expected total cost. We provide details about the RMP and SP formulations in Appendix A.2.2.

We use the problem sets in Crainic et al. (2014), i.e., five problem sets (IV–VIII) from the set of R instances in Crainic et al. (2011). Each problem set uses the same network, with parameters of each network shown in Table 2.2. The instances were originally proposed for the deterministic fixed charge multi-commodity network design problem (Crainic et al. 2001). We apply techniques in Song et al. (2014) to generate random samples. The demand $\tilde{v}_k$ of commodity k in each scenario ω follows a Normal distribution with mean equal to the demand in the deterministic instances and standard deviation equal to 0.1–0.2 times the mean.

Table 2.2: Instance Attributes of CMND

Problem Set	$\|N\|$	$\|A\|$	$\|K\|$
IV	10	60	10
V	10	60	25
VI	10	60	50
VII	10	82	10
VIII	10	83	25

2.4.2 Performance of the SVM Classifier

We first take Instance cap41 of CFLP and Instance r082 of CMND as examples to demonstrate the performance of the SVM classifier. We present the prediction accuracy for cuts in each data set in Table 2.3. For each instance, we present the accuracy of five sets. The first set is the training data

set, which contains the cuts sampled in Phase 1. We create four validation data sets consisting of unseen cuts. To compute the prediction accuracy, we need to compute the "true labels" of cuts in the validation set by the label transformation function (2.15). Therefore, all the cuts in validation sets are collected in the same way as that of Phase 1 (see Algorithm 2.1), but using different parameters (as shown in Table 2.3). The validation data set 1 shares the same parameter as those of the training data set. The validation data set 2 uses the twice standard deviation as that of the training data set to generate realizations of the uncertain parameters for the optimization model in different scenarios. The validation data set 3 doubles the number of sampling paths, which can be equivalently viewed as a set sharing the same property with validation data set 1 but with a twice larger size. The validation data set 4 uses a larger length of the sampling path, which can be viewed as a set containing more types of cuts, i.e., the validation data set 4 also includes cuts generated in later iterations in addition to the cuts in the training data set.

Table 2.3: Prediction Accuracy of the SVM Classifier

Inst.	Prediction Accuracy (%)									
	Training Data Std $= 0.1$		Validation Data 1 Std $= 0.1$		Validation Data 2 Std $= 0.2$		Validation Data 3 Std $= 0.1$		Validation Data 4 Std $= 0.1$	
	$K = 2$	$K = 2$	$K = 2$	$K = 2$	$K = 2$	$K = 2$	$K = 4$	$N = 200$	$K = 2$	$N = 300$
cap41	99.78		78.25		67.50		75.00		78.33	
r082	98.15		82.95		74.90		83.57		82.10	

The results in Table 2.3 show that the in-sample prediction accuracy is higher than 98% while the out-of-sample prediction accuracy of the validation data sets is relatively lower. The out-of-sample prediction accuracy of validation data sets 1, 2, and 4 is at similar levels higher than 75% for both instances, and the prediction accuracy of Instance r082 is higher. The out-of-sample prediction accuracy of validation data set 2 is the lowest among the validation sets. This is because the uncertain model parameters across scenarios of the validation data set 2 are different from that of the training data set, and thus the generalization power is weaker than other validation sets. Please note that the "true labels" of the validation data sets are computed by (2.15), which are our belief but not the exact classification of valuable or non-valuable cuts.

2.4.3 Results of Comparing LearnBD with Benders Decomposition

As mentioned in Section 2.1.2, the main obstacle of the traditional Benders is the large size of RMPs and, consequently, the long CPU time spent on solving RMPs in each iteration. SPs have smaller sizes and linear programming structures, and they can be efficiently solved in parallel. Thus, the total time for solving a testing problem with Bender's approach is almost the cumulative

time for solving RMPs. Therefore, we refer to algorithm efficiency as the RMP solving time in the rest of the paper. In this section, we present the computational results for solving the CFLP and CMND instances with traditional Benders decomposition (BD) and LearnBD. Specifically, we show the number of iterations, optimality gap, number of cuts, and cumulative time of solving RMPs (i.e., the last four columns in Table 2.4, Table 2.6, and Table 2.7). For LearnBD, we create a training problem for collecting cut information and we re-use this information when solving the testing problems. For all instances, we set the initial value of Δ as 1.2 and decrease it by 0.01 for each retraining conducted, i.e., $\mathcal{L}_\Delta = [1.2 - 0.01i]$ for $i = 0, \ldots, 50$.

2.4.3.1 Results of CFLP.

We compare the results of solving CFLP instances using LearnBD and BD in Table 2.4. Based on the problem size and the computational difficulty, we set the precision parameter $\delta = 0.01\%$ and the time limit as one hour. The solving process terminates when $\frac{\text{UB-LB}}{\text{LB}} < \delta$ (see Step 4 in Algorithm 2.2) or the cumulative solving time of the RMP reaches the time limit.

Table 2.4: Results of CFLP Instances Solved by LearnBD and BD

Problem Set	Instance	Std_Training (× mean)	$\lvert\Omega\rvert$	Std_Testing (× mean)	Method	Number of Iter.	Opt Gap (%)	Number of Cuts	Total Time of RMPs (s)
IV	cap41	0.1	100	0.1	BD	40	–	4000	183.48
					LearnBD	38	–	3790	96.31
				0.2	BD	47	–	4700	187.05
					LearnBD	45	–	4467	141.30
	cap42	0.1	200	0.1	BD	30	–	6000	111.96
					LearnBD	32	–	4304	84.28
				0.2	BD	30	–	6000	93.93
					LearnBD	29	–	5796	86.72
	cap43	0.1	400	0.1	BD	27	–	10800	243.57
					LearnBD	26	–	10378	220.18
				0.2	BD	26	–	10400	228.66
					LearnBD	26	–	10369	215.07
	cap44	0.1	400	0.1	BD	24	–	9600	235.34
					LearnBD	25	–	9938	156.52
				0.2	BD	22	–	8800	172.99
					LearnBD	22	–	8764	134.11
VI	cap62	0.1	50	0.1	BD	258	3.26	12900	LIMIT
					LearnBD	215	2.54	10701	LIMIT
				0.2	BD	206	3.06	10300	LIMIT
					LearnBD	216	2.53	10786	LIMIT

In Table 2.4, (i) all instances in problem set IV can be optimized within the one-hour time limit.

For these instances, the cuts generated by LearnBD are fewer than those of BD; the time of solving RMPs in LearnBD is shorter, and the total time reduction ranges from 6% (for cap43-0.2) to 47.5% (for cap41-0.1). (ii) Instance cap62 is much more difficult to solve compared with other instances in problem set IV. Both BD and LearnBD exceed the one-hour time limit and LearnBD has a smaller optimality gap within one-hour time limit. The number of iterations and cuts of LearnBD are more than those of BD. It happens that LearnBD executes more iterations because it adds fewer cuts than BD during each iteration. For the second testing problem of the Instance cap62, LearnBD can solve RMPs with more cuts, which implies that the RMPs of LearnBD is easier to solve than the ones in BD. In Table 2.5, we present the time and the number of cuts needed by LearnBD to reach the same or smaller optimality gap of BD shown in Table 2.4 for solving Instance cap62. Demonstrated in Table 2.5, LearnBD adds fewer cuts and takes much shorter time to achieve similar optimality gap as BD for the hard instance. (iii) For each instance, by comparing testing problems with two different standard deviations used for generating demand scenarios, we conclude that the improvement of LearnBD is more significant for testing problems having similar uncertainty distribution as the training problem. One reason is that, for testing problems with larger standard deviation, the collected cuts in the training data in Phase 1 can be viewed as a subset of the total cut population that LearnBD encounters in Phase 2.

Table 2.5: Comparing LearnBD with BD for Solving Instance cap62 to Similar Accuracy

Problem Set	Instance	Std_Training ($\times$ mean)	$\lvert\Omega\rvert$	Std_Testing ($\times$ mean)	Method	Number of Iter.	Opt Gap ($\%$)	Number of Cuts	Total Time of RMPs (s)
VI	cap62	0.1	50	0.1	BD	258	3.26	12900	>3600
					LearnBD	168	3.12	8357	1851.10
				0.2	BD	206	3.06	10300	>3600
					LearnBD	194	3.05	9686	2674.99

2.4.3.2 Results of CMND.

We present the results of LearnBD and BD for solving CMND instances in Table 2.6, where the optimality tolerance $\delta = 1\%$ and the time limit is set as two hours.

In Table 2.6, most CMND instances take longer time than CFLP instances. We have similar observations as in Table 2.4: (i) Instances r046, r054, and r076 can be optimized within the two-hour time limit and the cumulative solving time of RMPs of LearnBD is significantly less than that of BD. The cuts in LearnBD are fewer than cuts generated by BD for Instances r046 and r076. (ii) The rest of instances are hard to solve and both LearnBD and BD reach the two-hour time limit before converging. Instance r071 is extremely hard to solve and the optimality gaps of the two testing problems are greater than 100% when reaching time limit. For all of these instances,

Table 2.6: Results of CMND Instances Solved by LearnBD and BD

Problem Set	Instance	Std_Training ($\times$ mean)	$\lvert\Omega\rvert$	Std_Testing ($\times$ mean)	Method	Number of Iter.	Opt Gap (%)	Number of Cuts	Total Time of RMPs (s)
IV	r041	0.1	80	0.1	BD	269	42.19	21520	LIMIT
					LearnBD	248	22.88	19761	LIMIT
				0.2	BD	275	20.44	22000	LIMIT
					LearnBD	244	18.18	19283	LIMIT
	r046	0.1	80	0.1	BD	32	–	2560	62.30
					LearnBD	28	–	1708	41.70
				0.2	BD	28	–	2240	111.53
					LearnBD	35	–	2094	80.81
V	r051	0.1	100	0.1	BD	185	9.79	18500	LIMIT
					LearnBD	155	3.66	15210	LIMIT
				0.2	BD	193	10.37	19300	LIMIT
					LearnBD	190	8.69	18803	LIMIT
	r054	0.1	100	0.1	BD	64	–	6400	585.944
					LearnBD	56	–	5398	385.89
				0.2	BD	85	–	8500	1172.40
					LearnBD	78	–	7041	984.71
VI	r061	0.1	100	0.1	BD	121	10.54	12100	LIMIT
					LearnBD	125	9.01	12288	LIMIT
				0.2	BD	137	10.28	13700	LIMIT
					LearnBD	135	9.69	13602	LIMIT
VII	r071	0.1	80	0.1	BD	159	958.69	12720	LIMIT
					LearnBD	170	884.19	13194	LIMIT
				0.2	BD	154	967.37	12320	LIMIT
					LearnBD	167	885.94	13215	LIMIT
	r075	0.1	80	0.1	BD	121	10.89	9680	LIMIT
					LearnBD	150	9.24	9621	LIMIT
				0.2	BD	119	11.23	9520	LIMIT
					LearnBD	126	10.07	9350	LIMIT
	r076	0.1	80	0.1	BD	38	–	3040	360.93
					LearnBD	52	–	3026	276.91
				0.2	BD	38	–	3040	374.81
					LearnBD	48	–	3080	305.71
VIII	r082	0.1	80	0.1	BD	106	9.77	8480	LIMIT
					LearnBD	113	9.27	8873	LIMIT
				0.2	BD	104	9.91	8320	LIMIT
					LearnBD	106	7.13	8156	LIMIT

LearnBD obtains a smaller optimality gap. In Table 2.7, we further show the computational time and number of cuts needed by LearnBD to achieve similar optimality gap as BD for the instances that were not solved to optimality either by BD or LearnBD in Table 2.6.

Table 2.7: Comparing LearnBD with BD for Solving CMND instances to Similar Accuracy

Problem Set	Instance	Std_Training ($\times$ mean)	$\|\Omega\|$	Std_Testing ($\times$ mean)	Method	Number of Iter.	Opt Gap (%)	Number of Cuts	Total Time of RMPs (s)
IV	r041	0.1	80	0.1	BD	269	42.19	21520	> 7200
					LearnBD	204	27.41	13241	4140.49
				0.2	BD	275	20.44	22000	> 7200
					LearnBD	158	20.39	12403	2505.91
V	r051	0.1	100	0.1	BD	185	9.79	18500	> 7200
					LearnBD	63	7.67	6010	249.52
				0.2	BD	193	10.37	19300	> 7200
					LearnBD	119	10.36	11764	2197.31
VI	r061	0.1	100	0.1	BD	121	10.54	12100	> 7200
					LearnBD	71	10.45	6902	1129.37
				0.2	BD	137	10.28	13700	> 7200
					LearnBD	70	10.26	6835	1030.30
VII	r071	0.1	80	0.1	BD	159	958.69	12720	> 7200
					LearnBD	143	937.63	11034	4868.40
				0.2	BD	154	967.37	12320	> 7200
					LearnBD	139	966.31	10975	4816.33
	r075	0.1	80	0.1	BD	121	10.89	9680	> 7200
					LearnBD	115	10.86	6821	2416.36
				0.2	BD	119	11.23	9520	> 7200
					LearnBD	103	11.19	7510	3641.27
VIII	r082	0.1	80	0.1	BD	106	9.77	8480	> 7200
					LearnBD	55	9.75	4233	590.09
				0.2	BD	104	9.91	8320	> 7200
					LearnBD	54	8.72	4158	658.68

In Table 2.7, LearnBD uses much shorter time to obtain a similar (slightly tighter) gap as the gap that BD achieves under the two-hour time limit for all instances. And similarly, LearnBD performs better in testing problems which have similar uncertainty distribution as the training problems. For instances r051 and r082, LearnBD achieves a similar optimality gap within 10% of the time spent by BD. Comparing the optimality gap results and the solving time of LearnBD versus BD in Tables 2.6 and 2.7, we notice that BD takes longer to attain solutions with similar optimality gaps as the ones of LearnBD for the very hard instances that cannot be optimized within the 2-hour time limit by either method in Table 2.6. Therefore, we conclude that for instances which are hard to solve: (i) when we allow a relatively large optimality gap, for example, 10% for Instance r051, LearnBD can achieve this gap in significantly shorter time compared to BD; and (ii) when we

allow a relatively small gap, both LearnBD and BD need several iterations and thus long solving time. Note that LearnBD only takes 249 seconds while BD takes more than 7200 seconds to solve Instance r051 to a less than 10% gap.

Remark 2.4.1. *Based on our computational results, retraining almost happens in every instance and Δ may decrease consecutively over iterations before the classifier can identify valuable cuts, i.e., we may consecutively perform retraining. As mentioned in Remark 2.3.6, the retraining can be done before the solving process and thus does not affect the total solving time. The label predicting process of cuts is extremely fast and can be implemented in parallel. Therefore, the time consumed by retraining, or consecutive retraining, is also negligible when considering the total solving time. In our computational tests, LearnBD decreases the values of Δ relatively more frequently in the first several iterations and then keeps using a fixed Δ until the termination. For readers' interest, we present the values of Δ during iterations of five CMND instances in Table 2.8.*

Table 2.8: Δ-values Throughout Computational Iterations of Five CMND Instances

r041	Iteration	1–3	3–248							
	Δ	1.20	1.13							
r051	Iteration	1–2	3	4–155						
	Δ	1.07	1.06	1.01						
r071	Iteration	1–5	6–170							
	Δ	1.20	1.12							
r075	Iteration	1	2	3	4–5	6–7	8–15	16–150		
	Δ	1.20	1.19	1.18	1.13	1.07	1.06	1.00		
r076	Iteration	1	2	3	4	5	6–7	8–10	11–15	16–52
	Δ	1.20	1.11	1.06	1.05	1.02	1.01	1.00	0.99	0.98

2.4.3.3 Results of Replicates for CFLP Instance.

In this section, we provide additional computational results for five replicates of each instance in the CFLP problem set IV to show the performance consistency of comparing LearnBD and BD across randomly sampled scenarios. The demand scenarios of five replicates of each instance are generated independently with the same standard deviation as the training problem. We present the minimum, maximum, mean, and median values of (i) the number of generated cuts and (i) the total solution time of RMPs in Table 2.9. The observations are consistent with those in Section 2.4.3.1 and Section 2.4.3.2 that LearnBD can save more time than BD for optimizing the testing problems.

Table 2.9: Results of Multiple Runs for CFLP Instance from Problem Set IV

Instance	Method	Number of Cuts				Total Time of RMPs (s)			
		Min	Max	Mean	Median	Min	Max	Mean	Median
cap41	BD	3000	4000	3740	3900	124.38	209.68	161.89	154.11
	LearnBD	2499	3790	3366.2	3446	81.08	173.07	106.55	96.31
cap42	BD	5800	6000	5920	6000	71.70	124.95	99.95	104.79
	LearnBD	4304	5798	5123.2	5177	63.10	100.56	78.61	81.39
cap43	BD	10400	10800	10640	10800	147.02	263.42	211.64	221.44
	LearnBD	10254	10486	10374	10378	145.02	246.86	200.44	214.07
cap44	BD	8000	9600	8720	8800	171.10	249.28	213.06	217.24
	LearnBD	8374	9938	8927.6	8774	135.97	174.92	159.06	156.52

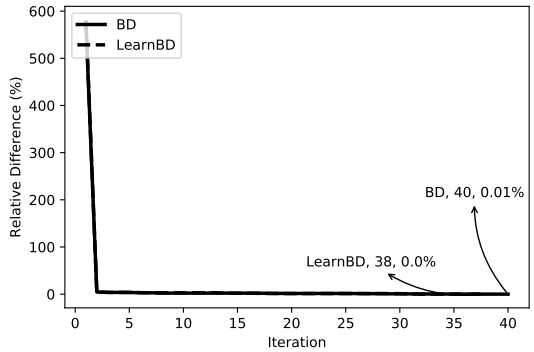
2.4.4 Performance Comparison Over Iterations

To track the performance of BD and LearnBD over iterations to show their convergence, we depict and analyze the results of two specific instances in Section 2.4.4.1 and Section 2.4.4.2. For each instance, we solve it with both BD and LearnBD separately to the same optimality gap. For each approach, we record the following values after each iteration:

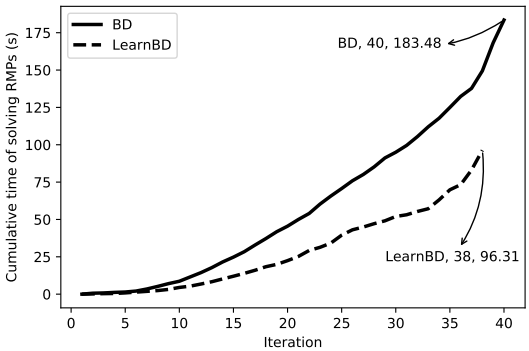
- the optimality gap after the current iteration, which helps track the algorithm convergence;
- the cumulative time for solving RMPs;
- the total number of cuts added to RMPs in the previous iterations, which reflects the size of RMPs and power of the classifier;
- the cumulative time for solving SPs, which can be performed in parallel in each iteration, and therefore this value will not affect the total solving time of the algorithm.

2.4.4.1 CFLP: Problem IV, cap41.

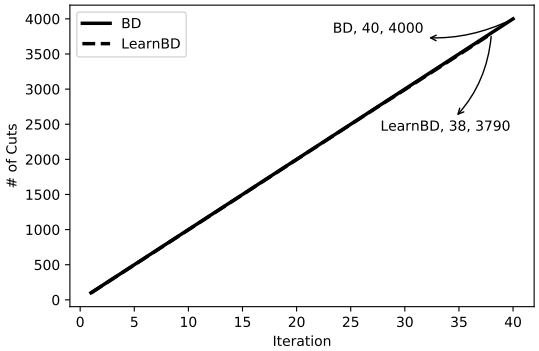
The results over the iterations are shown in Figure 2.2. The instance has 100 scenarios and the termination criterion is reaching $\delta = 0.01\%$ optimality gap. BD requires 40 iterations while LearnBD takes 38 iterations. LearnBD adds fewer cuts but achieves a similar optimality gap, which indicates the power of our SVM cuts classifier. Due to the smaller sizes of RMPs, the cumulative time for solving RMPs in LearnBD is significantly shorter than that in BD.



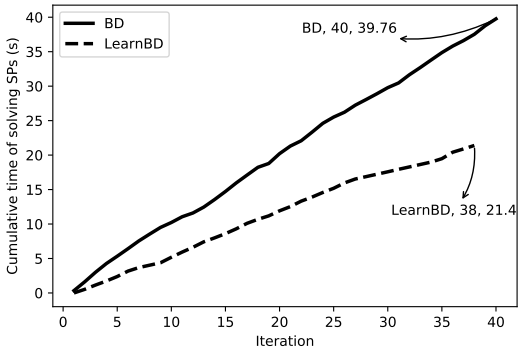
(a) Optimality gap.



(b) Cumulative time for solving RMPs.



(c) Number of cuts added to RMPs.

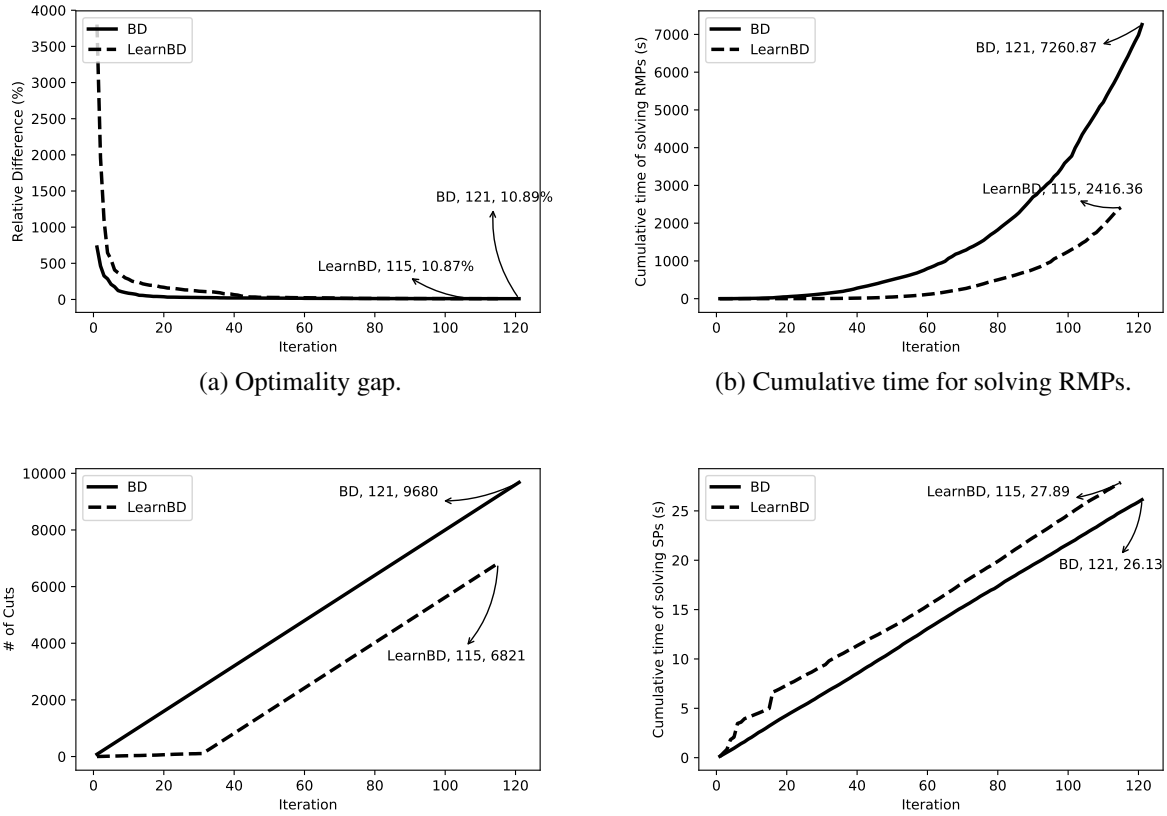


(d) Cumulative time for solving SPs.

Figure 2.2: CFLP: Problem set IV, cap41 instance solved by BD and LearnBD. The horizontal axis is the iteration number.

2.4.4.2 CMND: Problem Set VII, r075.

The results over the iterations for both BD and LearnBD are shown in Figure 2.3. The instance has 80 scenarios and the termination criterion is reaching $\delta = 10.89\%$ optimality gap (the result that BD achieves within the two-hour time limit). We observe similar performance of LearnBD as in the case of solving the CFLP instance. In Figure 2.3a, in the first 40 iterations, the gap of LearnBD convergences slower than BD because it adds fewer cuts. After 100 iterations, both algorithms achieve similar gaps. In Figure 2.3c, in the later iterations, the number of added cuts per iteration is quite similar to that added by BD. Thus, we can conclude that the first few iterations are important for reducing the total solving time.



(a) Optimality gap.

(b) Cumulative time for solving RMPs.

(c) Number of cuts added to RMPs.

(d) Cumulative time for solving SPs.

Figure 2.3: CMND: Problem set VII, r075 instance solved by BD and LearnBD. The horizontal axis is the iteration number.

2.4.5 Sampling and Training Time

As we mentioned in Remark 2.3.2 and Remark 2.3.6, the cut sampling and classifier training processes can be done before starting solving the testing problems. The outcomes of these two processes can be repeatedly used for solving different testing problems. Therefore, the time of these two processes does not affect the total solving time of LearnBD. Here, we record and present the time of Phase 1 and the time of classifier training in Phase 2 in Table 2.10. Column K shows the number of sampling paths and Column N shows the length of the sampling paths, i.e., the number of iterations in each sampling path, in Phase 1. We include the following four columns showing the results of solving one testing problem of each instance: Method, Opt Gap, Number of Cuts, and Testing-Time. Note that the testing problems are different from the training problem and for each instance, we only include results of one testing problem with the same standard deviation as the training problem. The last two columns in Table 2.10 are: "Phase-1-Time" showing the total time of cut sampling and collecting information from training problems in Phase 1 and "SVM-Training-Time" showing the total time used for training and retraining SVM classifiers.

From Table 2.10, the time of Phase 1 is highly dependent on N, while the time of each sampling path is the cumulative time for solving N number of RMPs with n cuts in iteration n for $n = 1, \ldots, N$. In our computational studies, we use $N = 2 \times |\Omega|$, i.e., proportional to $|\Omega|$, and thus the time in Phase 1 is highly related to $|\Omega|$. In most of the instances, the time in Phase 1 and the training time of SVM classifiers are relatively short as compared to the total solving time. Together with the results in Table 2.7, for Instances r051, r061, and r082, LearnBD reduces the total solving time by 80% to reach a 10% optimality gap even if LearnBD conducts Phase 1 for solving only one testing problem (if we also consider the time in sampling cuts and training classifiers).

2.4.6 Classifier Transfer Between Instances

Previously, we propose to solve a given two-stage stochastic optimization problem with cuts sampled from a similar training problem, where the only difference between the training problem and the original problem is the underlying distribution of the uncertain parameter. Intuitively, if the problem can be solved with training data collected from another training problem in a different size, e.g., the different number of variables and constraints, then the algorithmic efficiency can be further improved because we can use the same training data, or equivalently speaking, we can transfer the classifiers to solve other problems.

In this section, we present the results where the training problem is different from the original problem as an extension. We consider two instances cap42 and cap62 of CFLP. We normalize the two cut characteristics, cut violation and number of cuts generated by the same scenario, to eliminate the incompatible effects of the difference between the training and testing problems. In-

Table 2.10: Sampling and Training Time

Instance	K	N	Method	Opt Gap (%)	Number of Cuts	Testing-Time (s)	Phase-1-Time (s)	SVM-Training-Time (s)
r041	2	80	BD	42.19	21520	> 7200	-	-
			LearnBD	22.88	19761	> 7200	37.63	0.01
r046	2	80	BD	1	2560	62.30	-	-
			LearnBD	1	1708	41.70	15.29	0.01
r051	2	100	BD	9.79	18500	> 7200	-	-
			LearnBD	3.66	15210	> 7200	22.39	0.01
r054	2	100	BD	1	6400	585.94	-	-
			LearnBD	1	5398	385.89	24.41	0.38
r061	2	100	BD	10.54	12100	> 7200	-	-
			LearnBD	9.01	12288	> 7200	28.59	0.40
r071	2	80	BD	958.69	12720	> 7200	-	-
			LearnBD	884.19	13194	> 7200	186.77	0.30
r075	2	80	BD	10.89	9680	> 7200	-	-
			LearnBD	9.24	9621	> 7200	58.94	0.22
r076	2	80	BD	1	3040	360.93	-	-
			LearnBD	1	3026	276.91	51.93	0.21
r082	2	80	BD	9.77	8480	> 7200	-	-
			LearnBD	9.27	8873	> 7200	45.26	0.30

stead of using the absolute value of cut violation, we scale the cut violation by $\frac{\sum_{j\in F}\bar{d}_j\bar{c}}{\bar{k}}$, where $\bar{d}_j$ is the nominal value of the uncertain demand of factory j, $\bar{c} = \frac{\sum_{i\in W, j\in F} c_{ij}}{|W|\cdot|F|}$ is the average transportation cost, and $\bar{k} = \sum_{i\in W} k_i/|W|$ is the average warehouse setup cost. The scalar $\frac{\sum_{j\in F}\bar{d}_j\bar{c}}{\bar{k}}$ can be reviewed as the relative total transportation cost, which reflects the magnitude of the optimal objective function value of the CFLP problem. The second cut characteristic, the number of cuts generated by the same scenario, is related to the required number of iterations if using the traditional Benders. This characteristic is hard to estimate before solving the problem. Thus, we propose to use the size of the transportation network, i.e., $|W| \cdot |F|$, to scale it.

The results are presented in Table 2.11, for which we set the precision δ as 0.01% and the time limit as one hour. In the first two rows, we solve cap42 with traditional BD and the proposed LearnBD where the training data is collected from a training problem with the same model parameters (the results are the same as that of cap42 in Table 2.4). In the third row, we construct a training problem from cap62 and then use the training data to train an SVM classifier and solve cap42.

Table 2.11: Results with Transferred Classifier

Inst.	$\|\Omega\|$	Std.Testing ($\times$ mean)	Method	Training Instance	Std. Training ($\times$ mean)	Number of Iter.	Opt Gap (%)	Number of Cuts	Total Time of RMPS (s)
cap42	100	0.1	BD	-	-	30	0.01	6000	111.96
			LearnBD	cap42	0.1	32	0.01	4304	84.28
			LearnBD	cap62	0.1	29	0.01	5794	97.96

In Table 2.11, LearnBD with a transferred SVM classifier, i.e., the third row, also reduces the cumulative time of RMPs as compared to BD. Via comparing the second and the third rows, LearnBD trained with the same instance adds fewer cuts and takes short time to solve RMPs. Therefore, we can conclude that with a proper scaling rule of the two cut features, the training data can also be re-used for solving other instances to improve the solving efficiency.

2.5 Concluding Remarks

In this paper, we developed a learning-enhanced Benders decomposition algorithm to accelerate the solving process of Benders decomposition, one of the most useful algorithms for solving two-stage stochastic programs. The bottleneck for traditional Benders decomposition is the increasing sizes and the long solving time of RMPs. We restricted RMP sizes over iterations by distinguishing valuable cuts. The computational studies based on capacitated facility location and multi-commodity network design instances demonstrated the power of SVM cut classifier. With a proper selection of

hyperparameters, the LearnBD algorithm worked efficiently with smaller sizes and shorter solving time of RMPs compared to traditional Benders decomposition.

Our numerical results of diverse instances showed that LearBD can achieve better computational performance than BD for solving different types of benchmark two-stage stochastic programs considered in the literature. We consider the following future research directions to improve our algorithm. First, we can extend the characteristics and performance indices for the current LearnBD algorithm to capture multiple types of information of cuts. The second direction is to explore the possibility of constructing an online learning algorithm using reinforcement learning, which requires decomposing the effects of multiple cuts added simultaneously into the same RMP. We are also interested in improving LearnBD for solving a broader range of large-scale problems with special structural properties.

CHAPTER 3

Context-aware Route Recommendation with Weight Learning through Deep Neural Networks

We consider real-time origin-destination transportation requests in, e.g., ride-hailing, and for each request, provide a context-aware route recommendation. To overcome the difficulty of estimating uncertain user preferences toward multiple route features, we design and implement an approach via a combination of the weighted shortest path problem and deep learning, and evaluate it using real-world transportation data. Specifically, the proposed approach learns weights from historical choices of drivers through a deep neural network by minimizing the total weighted costs of historical routes and maximizing those of the non-chosen routes. We evaluate our method and two benchmarks with 4 million requests received by Didi Chuxing in Beijing. Based on the results, we demonstrate that distinguishing request scenarios helps provide preferable context-aware route recommendations.

3.1 Introductory Remarks

Consider real-time origin-destination (O-D) transportation requests in, e.g., map navigation and ride-hailing. The requests are originated by drivers, who are later provided with one route for each request. For ride-hailing companies, such as Uber and Didi Chuxing, route recommendation is one of the critical business elements, and they aim to improve the effectiveness of the recommendation system. Recommended routes that are not consistent with user preferences may be abandoned, leading to low user satisfaction rates, and eventually causing drivers or riders to switch to other service providers. On the other hand, user preferences present substantial heterogeneity and they are not explicitly known when planning routes (see, e.g., Yang et al. (2015)).

Efficient algorithms were developed in the field of Operations Research for solving route planning problems involved in vehicle routing (see, e.g., Laporte (1992), Shen et al. (2017), Song and Shen (2016), Geisberger et al. (2008)) and risk-averse shortest path (see, e.g., Bell (2006), Gavriel

et al. (2012)). We typically formulate route planning as a weighted shortest path problem (WSPP). In WSPP, we consider a route evaluation problem where the cost is computed by a weighted sum of multiple route features. Given a route feature set K and a driver preference weight vector w_k for $k \in K$, a route with feature value f_k for $k \in K$ has a cost $\sum_{k \in K} w_k f_k$. As a result, we seek a route with the minimum weighted cost.

In this paper, we assume that we already have a set of candidate routes, which are generated by state-of-the-art route planning algorithms or combined from historical trajectories, and then we evaluate them and make context-aware recommendations considering the uncertain preferences of specific drivers and request scenarios. The obstacle of a WSPP-based evaluation system for practical utilization is the uncertain and dynamic weight vector. To tackle this problem, we propose a learning-based WSPP evaluation system. The system performs three tasks: (i) distinguishing different service contexts; (ii) estimating the weight vectors of drivers; (iii) and providing context-aware route recommendations. We characterize each travel request by its specific contextual information. Moreover, we include characteristic vectors of the drivers as part of the contextual information, and thus the system also provides personalized recommendations simultaneously. The contextual information includes both numerical and categorical values, and the relationship between contextual information and the weight vectors can be extremely complex. Therefore, we proposed to construct a deep neural network for estimating the weight vector. We conduct the following work. (i) Identify and collect contextual information and route features. (ii) Construct a deep neural network to estimate weights from request contextual information. (iii) Streamline a route evaluation procedure based on WSPP and estimated weights. (iv) Incorporate the evaluation results and the historical records and train the neural network. We elaborate on the details as follows.

3.1.1 WSPP

We construct a directed network $G(N, A)$ from a real-world road map, where nodes in set N represent geographical points and arcs in set A represent road segments that connect nodes in N. Features of arcs, such as travel time and distance, are included in the feature set K. For each feature $k \in K$ of arc $(i, j) \in A$, we evaluate its value f_{ijk}, where smaller value is more preferable, and assign a fixed weight w_k. A travel request is associated with an O-D pair, where $O, D \in N$. Let x_{ij} be a binary decision variable where $x_{ij} = 1$ denotes that arc $(i, j) \in A$ is included in the routing solution, and 0 otherwise. We show an example of WSPP in Figure 3.1.

If w_k is fully known for each feature $k \in K$, we can solve the WSPP through the following

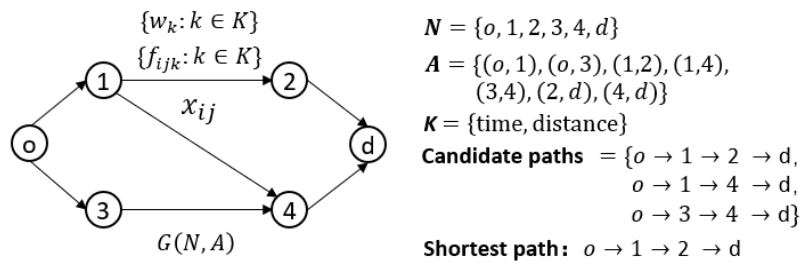


Figure 3.1: An example of WSPP on $G(N, A)$

binary linear program.

$$\text{(WSPP)} \quad \min_{x} \sum_{k \in K} w_k \sum_{(i,j) \in A} f_{ijk} x_{ij} \tag{3.1}$$

$$\text{s.t.} \quad \sum_{j:(o,j) \in A} x_{oj} - \sum_{j:(j,o) \in A} x_{jo} = 1, \tag{3.2}$$

$$\sum_{j:(i,j) \in A} x_{ij} - \sum_{j:(j,i) \in A} x_{ji} = 0, \; i \in N \backslash \{o, d\}, \tag{3.3}$$

$$\sum_{j:(d,j) \in A} x_{dj} - \sum_{j:(j,d) \in A} x_{jd} = -1, \tag{3.4}$$

$$x_{ij} \in \{0, 1\}, \quad (i, j) \in A. \tag{3.5}$$

The objective function in (3.1) is a weighted sum of all feature values of arcs in the routing solution. Constraints (3.2) and (3.4) restrict that the route starts and ends at origin and destination nodes, respectively. Constraint system (3.3) indicates that if the solution goes to one intermediate node and then it will leave this node too (which is also referred to as a flow balance constraint). Constraint (3.5) requires that all the x-variables are integer-valued. In this basic WSPP, all route features are summable over route segments. Note that due to the constraint matrix is totally unimodular, we can further relax constraints (5) as $x_{ij} \geq 0, \; \forall (i, j) \in A$ and obtain binary-valued solutions at optimum (see, e.g., Ahuja et al. (1993)).

3.1.2 Literature Review of Context-aware Route Planning

To evaluate uncertain preferences of drivers towards multiple objectives, researchers distinguish contexts and model route choice behaviors in different contexts (see, e.g., Zangui et al. (2013), He et al. (2014a), Lou et al. (2010), Xu et al. (2011)). Refs. Yang et al. (2015) and Dai et al. (2015) define procedures for computing context representation vectors from historical trajectory data of drivers and then provide customized route planning solutions over connected road networks, where the latter also includes a graph contraction to improve route planning efficiency. Ref. Campigotto et al. (2016) defines a set of user profile categories, and for a new request, it provides a customized

route recommendation by distinguishing the context category of the user. The authors also develop a Bayesian algorithm to update the profiles of users. Ref. Wang et al. (2018) defines a driver habit vector and identifies requests with similar contexts by a log fuzzy C-means clustering algorithm. The authors implement a navigating algorithm from a previous similar trip group to a specific individual transportation request.

3.1.3 Contributions

We summarize the main contributions as follows.

- We develop a learning-based approach to extract the weight knowledge associated with each route feature from the historical transportation data. There is no record of weights in each request, and thus it is impossible to directly compute the weight distribution through statistical approaches and then apply stochastic programming models. The weight learning approach overcomes this difficulty and provides insights for problems where uncertain parameters cannot be sampled and estimated directly.
- We bridge the learning approach and optimization problem by customizing the loss function of the deep neural network using objective values of WSPP. The definition of the loss function makes the training process meaningful and also the outputs of the network, which serves as weights, are more consistent with WSPP.
- Computational studies are conducted on large-scale real-world data received by Didi Chuxing in Beijing, involving about 4 million requests in total. We estimate the percentage of drivers who would follow our recommended routes, to demonstrate the performance of the proposed learning-based route recommendation system versus two benchmarks.

3.1.4 Structure

In Section 3.2, we describe the problem and notations in detail. Specifically, we show an intermediate data processing step for extracting historical preferences of drivers as part of the contextual information. In Section 3.3, we construct a Learning-based Context-aware Route Recommendation System (LCRRS) to learn the weight through deep learning and define a specific loss function based on the objective function of WSPP to train the network. In Section 3.4, we present computational results on a real-world dataset to demonstrate the efficacy of the proposed LCRRS. In Section 3.5, we conclude this paper and discuss future research directions.

3.2 Problem Description and Notations

We consider providing a context-aware route recommendation to a transportation request with a given O-D pair. Each O-D request comes with a set of candidate routes among which we perform route evaluation by considering the contextual information and associated route features. For each request, we collect contextual information, which characterizes its context and helps estimate weights associated with different route features. A deep neural network is deployed to estimate weights, trained from historical route choices. Therefore, for historical request data, we also collect a similarity score of each candidate route as a performance index. This performance index indicates the true routes that drivers prefer and also the ranking scores of other candidate routes. To summarize, the historical data we have is request-wise, including contextual information, candidate routes with corresponding features, and similarity scores, defined as follows.

Definition 3.2.1. ***Contextual Information.*** *Contextual information is request-related but independent of route selections, such as origin and destination locations, departure time, type of services, driver history (see details in the later Definition 3.2.5), and point-of-interest category (POIC) of destination. The contextual information defines the request service scenario and affects route selection.*

Definition 3.2.2. ***Candidate Route.*** *We plan to learn weights associated with route features from the historical choices of drivers from candidate sets. In historical data, each request is accompanied by a set of candidate routes. Each candidate route is an ordered sequence of links in the road network connecting the origin and the destination with intermediate links.*

Definition 3.2.3. ***Route Feature.*** *Some route features are independent of requests, and we call them static route features, e.g., the length of the route, number of turns, and the combination of different types of road segments. The remaining features can vary among different requests, and we call them dynamic route features, e.g., estimated travel duration and traffic speed. We assume that all the static route features are accurately collected and all the dynamic route features are confidently estimated.*

Definition 3.2.4. ***Similarity Score.*** *Trajectory of a request is a sequence of locations with timestamps while the true route is a sequence of links in the road network that is mapped from the true trajectory. The published historical data does not include true routes out of the protection for private information, but each candidate route is assigned a score between* $[0, 1]$ *showing the similarity to the true route. The similarity score is computed as the intersection over union of the route length:*

$$\textit{Similarity score} = \frac{\textit{length}(\textit{true route} \cap \textit{candidate route})}{\textit{length}(\textit{true route} \cup \textit{candidate route})}.$$

For each request, at most one candidate route has 1*, which means that this candidate route is selected by the driver. We assume that at least one high-quality route is included in the candidate set of routes, and thus, the true route selected by the driver is in the set. This assumption is made without loss of generality by only utilizing this type of requests in the training and the testing processes.*

Definition 3.2.5. ***Driver History.*** *To achieve context-aware route recommendations, we introduce driver preference towards a subset of important route features as complementary contextual information. First, we choose a subset* K' *of the route features. Preferences of drivers towards those features can be different and eventually affect route selections. A preference vector is computed for each driver based on his/her historical requests. For driver* u *with a set* O_u *of historical requests, the preference vector* $q^u \in \mathbb{R}^{|K'|}$ *contains:*

$$q_k^u = \sum_{o \in O_u} \frac{f_{o,k}^{min}}{f_{o,k}^{*}} \text{ for } k \in K', \tag{3.6}$$

where $f_{o,k}^{*}$ *is the value of feature* k *of true route (i.e., the candidate route with similarity score* 1*) of request* o *and* $f_{o,k}^{min}$ *is the minimum value of the feature* k *among all candidate routes of request* o*. For feature* k*, among all candidate routes the best we can achieve is* $f_{o,k}^{min}$*. Therefore, the larger the right hand side of* (3.6) *for a feature* k*, i.e., the smaller difference between* $f_{o,k}^{*}$ *and* $f_{o,k}^{min}$*, the higher the driver's preference to this feature. We can store this driver's historical preference separately and update it once a new request arrives. When adding this vector into contextual information of requests, we implement a linear normalization for each driver by forcing the sum of all preferences to be* 1*.*

3.3 Learning-Based Context-aware Route Recommendation System (LCRRS)

A context-aware route recommendation should satisfy two requirements. (i) It should take real-time traffic conditions into consideration and dynamically adjust recommendations. (ii) It should distinguish different preferences of drivers and provide routes that meet their preferences for each specific request. We treat a route recommendation as a WSPP. Therefore, those two requirements are satisfied when we have an accurate estimation of route feature values and their corresponding weights. The real-time traffic condition can be well estimated nowadays through sensors and GPS systems. However, it is hard to know drivers' preferences over route features. Although drivers make route selections according to their preferences, they cannot explicitly quantify preferences over route features. Furthermore, their preferences can change according to scenario contexts.

One solution is to summarize driver preferences from their historical routing decisions while distinguishing request contexts. In this paper, we develop an approach that can estimate driver preferences from historical data, and then we recommend routes with the lowest cost to drivers based on solutions to WSPP.

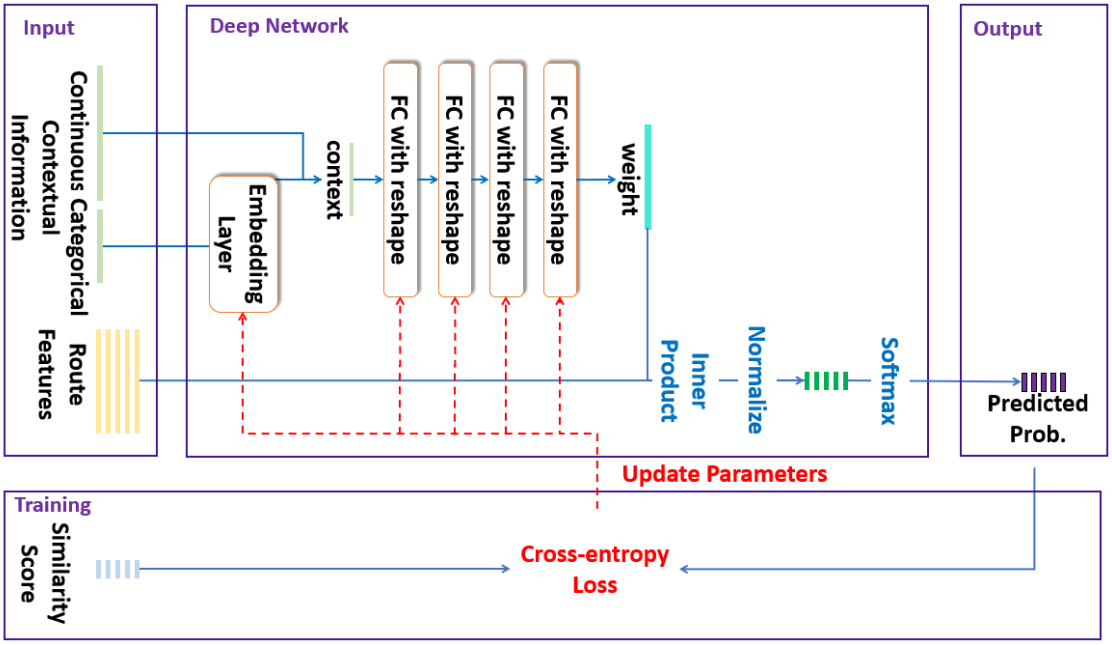


Figure 3.2: Overview of LCRRS

We develop a learning-based context-aware route recommendation system (LCRRS) to provide a route recommendation among a set of candidate routes. The system is incorporated with a deep neural network, which encodes the contextual information into weights and then outputs the context-aware route recommendation. We show an overview of LCRRS in Figure 3.2. For each request $o \in O$, we input contextual information $g^o = [g_v^o, g_c^o, q^u]$ and route features into LCRRS, and then it will output probabilities of all candidate routes to be the best choice for this specific request. In contextual information, g_v^o is continuous, e.g., departure time and the Euclidean distance between origin and destination; g_c^o is categorical, e.g., POIC of destinations; and $q^u \in \mathbb{R}^{|K'|}$ is the historical preference of driver u of request o, defined in (3.6). The contextual information passes through multiple hidden layers and is encoded into feature weights. Before going through multiple layers, categorical contextual information is embedded firstly and then concatenated with other continuous contextual information. Then, the learned feature weights are multiplied by their corresponding feature values for all candidate routes to compute the total costs of routes. We train LCRRS with the help of similarity scores of historical requests, which can be viewed as another type of ground truth of probabilities.

Embedding Layer Consider one categorical input with M categories, the input of an embedding layer is an M-dimensional vector. For the ith category, the corresponding input is a vector with 1 in the ith position and all 0s in other entries. Following a batch training framework, we manipulate N requests together each time, and thus, the input is $x \in \mathbb{R}^{N \times M}$, where N is the batch size. This input form is also referred to as the one-hot form of categorical data in Machine Learning literature (see, e.g., Santoro et al. (2016), Feurer et al. (2015)). An embedding layer is parameterized by a coefficient matrix $W \in \mathbb{R}^{M \times d_E}$, and the value of the matrix is obtained through the training process. The output is $y = xW^T \in \mathbb{R}^{N \times d_{embed}}$, and therefore, E is called embedding dimension. To conclude, each categorical information is embedded as an E-dimensional vector. In general, $E << M$, and one advantage of using an embedding layer is to reduce the dimension of the problem and improve computational efficiency. Another advantage of embedding is to introduce relations between categories, such as the distance of the embedded vectors (see, e.g., Mikolov et al. (2013)).

Fully Connected Layer (FC) with Reshape Given an input $x \in \mathbb{R}^{d_{in}}$, a coefficient matrix $W \in \mathbb{R}^{d_{in} \times d_{out}}$, and a bias vector $b \in \mathbb{R}^{d_{out}}$, the output h of the FC layer is defined as:

$$h(x) = \sigma(Wx + b), \tag{3.7}$$

where σ is an activate function to introduce nonlinearity and improve computational sparsity (see, e.g., LeCun et al. (2015)). We implement several activate functions and choose Rectified Linear Unit (ReLU), which has the best testing performance:

$$\text{ReLu}(x) = \begin{cases} 0, \text{ if } x \leq 0; \\ x, \text{ otherwise.} \end{cases} \tag{3.8}$$

The output of a d_{in}-dimensional input vector is a d_{out}-dimensional vector, and thus we call it a FC Layer with reshape. The input of the first FC layer is contextual information g_o. The output of the last FC layer (in the same dimension as route features) is an abstract representation of the request context, which can be viewed as feature weights.

Route Recommendation After we obtain a point estimation of the weight through learning, we estimate the total cost of each route by computing the inner product between weights and route features, following the framework of WSPP. We denote all candidate routes of a specific request o as $\mathcal{R}^o$ and denote the total cost, i.e., the objective value of WSPP in (3.1), of a route $r \in \mathcal{R}^o$ as s_r^o. We normalize the total costs of candidate routes and predict the probability of route r to be the

best choice of request o through a softmax function:

$$p_r^o = \frac{e^{s_r^o}}{\sum_{r' \in \mathcal{R}^o} e^{s_{r'}^o}}, \text{ for } r \in \mathcal{R}^o,\ o \in O. \tag{3.9}$$

That is, we input the contextual information and route features to LCRRS and recommend the route with the highest probability.

Cross-entropy Loss Function We assume that drivers made their route choices after fully evaluating candidate routes. We compute the objective value, as defined in (3.1), of the chosen route with learned weights. We expect that it is minimized or equivalently, the predicted probability is maximized. To facilitate the training process, we aim to achieve the opposite for all non-chosen routes, i.e., the probabilities of non-chosen candidate routes are minimized. To balance those losses coming from different routes, similarity scores are employed in training data and we construct the loss function following with a similar structure to cross-entropy loss function:

$$Loss(\mathbf{W}, \mathbf{b}) = \sum_{o \in O} \left[\sum_{r \in \mathcal{R}^o, r \neq r^*} \log(l_r^o e^{p_r^o}) - l_{r^*}^o \log(e^{p_{r^*}^o}) \right], \tag{3.10}$$

where $(\mathbf{W}, \mathbf{b})$ are parameters of all hidden layers, r^* is the true route of request o, and l_r^o is the similarity score of candidate route $r \in \mathcal{R}^o$. In the training process, we firstly initialize coefficient matrices $\mathbf{W}$ with values sampled from random normal distributions and initialize bias vectors $\mathbf{b}$ to zero vectors. Then we can train parameters iteratively using a mini-batch gradient descent approach by back-propagation.

3.4 Computational Studies

To evaluate the performance of LCRRS, we test it on real-world historical request data. In Section 3.4.1, we introduce training and testing. In Section 3.4.2, we define a lower and an upper bound of acceptance ratio as a performance index to evaluate LCRRS. In Section 3.4.3, we increase contextual information sequentially and show the results compared with two benchmarks. In Section 3.4.4, we present two implementation variants to handle more complicated application scenarios and show their testing results.

3.4.1 Data Collection

We implement our model on a real-world dataset collected from Didi Chuxing, the world's leading mobile transportation platform. The dataset contains requests in Beijing from May 27 to June 2

in 2019. There are over $800,000$ request data each day, and among them, $500,000$ requests select their routes from candidate routes. The number of candidate routes of each request varies from 2 to 20, with an average of 5. The recorded candidate routes are generated by route planning algorithms, such as Contraction Hierarchy (CH, see Geisberger et al. (2008)), and combined from historical routes with high utilization frequencies. The company evaluated all candidate routes and then provided one route for each request. If a driver does not prefer the recommended route, he/she will actively choose another one from a list of candidate routes provided by Didi Chuxing or choose their own routes. We evaluate 31 contextual information and 89 route features. Finally, we include 26 types of contextual information and 31 route features in our system, where POIC is the only categorical contextual information passing an embedding layer, and we compute a 10-dimensional driver history. We randomly split the dataset such that 80% is used as training data, and 20% is used as testing data.

3.4.2 Lower and Upper Bounds of Acceptance Ratio

For ride-hailing or map navigation scenarios, it is common that drivers have no preference for routes before they submit requests. Therefore, the goal of our system is not to identify the best choice for a specific request but to recommend routes that drivers would like to follow. However, it is impractical to interact with drivers directly through apps and record the acceptance ratio (AR), i.e., the percentage of drivers who follow the recommendation. Therefore, we estimate a lower and an upper bound of AR over historical data, where we know the historical recommended routes to drivers. Specifically, we compute the following values:

1. N_{total}: the total number of testing requests.

2. N_{yes}: the number of requests where the true routes are ranked as the best by LCRRS. For those requests, if we provide the best routes identified by LCRRS, drivers will accept the recommendations.

3. N_{no}: the number of requests where (i) the best route evaluated by LCRRS was exactly the route recommended to the driver but (ii) the driver chose another route. For those requests, if we provide the best routes identified by LCRRS, the driver will not accept and will choose a different route.

We have $N_{total} = N_{yes} + N_{no} + N_{unknown}$, where $N_{unknown} \geq 0$ represents the requests where the best routes evaluated by LCRRS, historical recommendations, and true choices of drivers are all different. For $N_{unknown}$, we cannot predict if our recommendations were provided, drivers would

follow it or not. The lower bound of the acceptance ratio (LAR) over historical data is defined as:

$$\text{LAR} = \frac{N_{yes}}{N_{total}}.$$

It represents that we can provide proper recommendations for as least this percentage of requests. The upper bound of the acceptance ratio (UAR) over historical data is defined as:

$$\text{UAR} = 1 - \frac{N_{no}}{N_{total}}.$$

where $\frac{N_{no}}{N_{total}}$ denotes the percentage of the requests which are surely unsatisfied with our recommendation.

3.4.3 Results with Different Contextual Information

For the 26 types of contextual information, we increase them sequentially to show the power of distinguishing contexts for context-aware route recommendations. We present the results of LCRRS with 4 different contextual inputs and two benchmarks in Table 3.1. The contextual information inputs are:

- CI_1: longitude and latitude of the origin and destination, and POIC of the destination.
- CI_2: ball-distance between the origin and destination and departure time.
- CI_3: driver and customer types, peak hour type, whether the origin or destination is an airport, and service type.
- CI_4: a driver history vector computed for the following 10 route features (i.e., $|K'| = 10$): length of level i roads with $i = 1, 2, 3, 4, 5$, total length, number of links, number of road type changes, number of traffic lights, and estimated duration.

The first benchmark (T-WSPP) is exactly the WSPP with one route feature, i.e., the average passing time analyzed from historical data. T-WSPP generates routes with the minimum total expected travel time. The second benchmark (NC) is a variant of LCRRS, where we include no contextual information. NC is equivalent to learn a uniform weight vector for all requests. In Table 3.1, T-WSPP has the worst performance, which indicates that including more route features helps evaluate route selections. From the results of NC and LCRRS, we can conclude that with more contextual information, both the LAR and UAR are increased and the performance of LCRRS is improved.

Table 3.1: Results with Benchmarks.

Model	Contextual Information	$\|CI\|$	LAR(%)	UAR(%)
LCRRS	CI_1	9	50.28	86.23
	$CI_1 \cup CI_2$	11	55.42	88.51
	$CI_1 \cup CI_2 \cup CI_3$	16	60.28	90.44
	$CI_1 \cup CI_2 \cup CI_3 \cup CI_4$	26	63.44	90.76
T-WSPP	-	-	39.59	67.32
NC	$\emptyset$	0	42.59	76.77

3.4.4 Results with Different Implementation Variants

We perform two structural adjustments as follows.

1. **Batch padding** The model we tested in the previous section requires that all requests in training and testing data have the same number of candidate routes. Therefore, we performed data manipulations accordingly before splitting so that all requests have 5 candidate routes. To improve training and predicting efficiency, we incorporate a batch padding trick. In each training and testing batch, we add virtual candidate routes with undesirable route features and 0 similarity score. With batch padding, the system can now handle requests with any number of candidate routes.

2. **Link embedding** We construct a new embedding layer taking link-ID as input. For a candidate route, we pass all link-IDs through this embedding layer and then add all outputs as a one-dimensional route feature. This link embedding layer is trained together with other layers. Compared with the summarized route features we include above, this new route feature carries information related to each specific link. This link-related feature is important, especially when several candidates share similar summarized features, but the road segments of one route are much preferable.

We present the results with 26 types of contextual information in Table 3.2. In the second line, LAR keeps stable while UAR increases more. One of the reasons is that after batch padding, the number of candidate routes varies from 2 to 20. Therefore, it is less likely for our recommendations to be the historical recommended routes for requests with more than 5 candidates. In the third line, LAR is further improved, indicating that we can offer proper recommendations for more requests.

Table 3.2: Results with Different Model Structures.

Model	LAR(%)	UAR(%)
Without batch padding	63.44	90.76
With batch padding	63.94	92.25
With batch padding and link embedding	67.93	90.85

3.5 Concluding Remarks

In this paper, we proposed a weight learning and context-aware route recommendation system. In particular, we evaluate candidate routes based on WSPP, where the uncertain weights are learned from the historical choices of drivers through deep learning. Based on real historical request data we collected from a ride-hailing company in China, we train the proposed recommendation system and test its performance. The computational results indicate that contextual information is helpful in providing context-aware route recommendations, and we can reach a high acceptance ratio.

An important future research direction is to explore the possibility of estimating an ambiguity set of the distributions of weights, rather than just a point estimation. With this ambiguity set, we can provide more robust route recommendations. To achieve this goal, we plan to design a sophisticated loss function incorporating the objective function of the distributionally robust optimization model so that we can compute the gradient and train the network.

CHAPTER 4

Online Learning and Pricing for Service Systems with Reusable Resources

We consider a price-based revenue management problem with finite reusable resources over a finite time horizon. Stochastically arrived customers request an exponentially distributed service time and may balk and renege given insufficient resource. The resource unit, upon completion of serving one customer, can be released to serve the next customer immediately. The arrival, service, balking and reneging rates all depend on the price being offered. In this paper, we assume that the firm does not know the mappings between these rates and prices, and thus it makes adaptive pricing decisions in each period based only on past sales to maximize the cumulative revenue. We propose three new multi-armed bandit (MAB) based learning algorithms for finding near-optimal pricing policies. Compared with prior pricing and multi-armed/linear bandits literature, the salient difficulties of this problem lie in (i) the unknown rate-and-price mapping information, (ii) the dynamic nature of reusable resources being committed over time, (iii) the transient behavior of the service system when the price changes, and (iv) unbounded and heavy-tailed sub-exponential distributions of observed random variables. Our proposed algorithms contain a Warm-up Phase to eliminate the heavy-tail effects, which are brought by the Poisson arrival and service processes of the M/M/c service systems, and a Learning Phase to identify the optimal price. Our algorithms separate the Learning Phase into successive operational batches and select a price using past sales collected in previous batches. The performance measure is cumulative regret, which is the difference between the revenue attained by our approach and by a clairvoyant optimal pricing policy under full distributional information. When the set of prices is finite, we propose two algorithms, termed Batch Upper Confidence Bound (BUCB) and Batch Thompson Sampling (BTS), and prove that the cumulative regret is $O(\sqrt{PT\log(T)})$, where T is the total number of time periods and P is the cardinality of the feasible price set. When the set of prices is continuous with linear dependence on a d_f-dimensional feature vector, we propose a Batch Linear Confidence Bound (BLinUCB) algorithm and prove that the cumulative regret is $O(d_f \log(T) \log\log(T)\sqrt{T})$. Both regret results match the lower bound up to a logarithmic factor. As an intermediate step of regret analysis, we

also develop a coupling analysis for analyzing the time for a queue to reach the steady state from an empty state or from a steady state under another set of system parameters. Our numerical experiments demonstrate and confirm the efficacy of the proposed BUCB, BTS, and BLinUCB algorithms.

4.1 Introductory Remarks

Over the past decades, revenue management problems have received extensive attention and proposed methods have achieved huge successes in application areas including airline management, electric power management, hotel management, healthcare system, advertising capacity management, car rental business, and call center services (see, e.g., Talluri and Van Ryzin 2006, Özer et al. 2012). The bulk of existing literature focuses primarily on settings with perishable resources in which the product once sold is permanently removed from inventory. There has been limited literature on settings with *reusable* resources that can be repeatedly used to serve customers.

Undoubtedly, revenue management with reusable resources finds a wide range of applications in today's economy, such as cloud computing services, ride-hailing services, car/bicycle rental services, project team management, and hospitality industry. In these applications, the firm is endowed with a finite capacity of reusable products. Each customer requests a product, uses it for some random time, and returns it to the firm, at which point the product can be used to serve other customers (see, e.g., Levi and Radovanović 2010, Chen et al. 2017, Rusmevichientong et al. 2020). For example, firms such as Amazon and Microsoft offer cloud computing services and users utilize virtual machines for a certain duration of time to finish their computational job requests (see, e.g., Kaewpuang et al. 2013, Püschel et al. 2015). The exact usage duration of a request in a cloud is not specified *a priori* and thus the departure of demand is also stochastic. The firm needs to dynamically decide what price to offer to users based on the current capacity utilization as well as the received requests (see, e.g., Doan et al. 2020). Another salient example arises from the use of on-demand service platforms, such as ride-hailing services including Uber and Lyft, where randomly arrived customers are dynamically paired with service providers (i.e., the resource) and occupy them for a random amount of time (Banerjee et al. 2015, Liu and Li 2017, Bimpikis et al. 2019). The pricing problem in such a two-sided ride-hailing platform also considers assortment simultaneously (see, e.g., Yan et al. 2020). The firm needs to decide what price to charge the riders and how to pay the drivers, as well as dynamic pairing of drivers and riders. More examples can be found in car/bicycle rental business (Oliveira et al. 2018) and parking facility management (Owen and Simchi-Levi 2018). In all the aforementioned applications, customers' willingness for using the service is affected by the price being offered; however, the decision-maker may not know the random demand distributions and how demand reacts to price changes at the beginning of the

planning horizon (see, e.g., Xu and Li 2013). Moreover, the revenue does not only depend on the number of customers but also on service duration of each customer.

This motivates us to consider a general class of revenue management problems with finite reusable resources under incomplete demand-price and service-time information. In our problem, the firm dynamically makes adaptive pricing decisions based on real-time information observed during the past operation, to maximize the total cumulative revenue. The customers arrive at the service system according to a Poisson process and they request a unit of the resource for an exponentially distributed consecutive time on a first-arrive-first-serve basis. If the resource is available at the time of arrival, the new customer is immediately admitted into the system and the service immediately starts without delay. The amount that a customer being charged is computed as the product of service time and offered price. If there is no available resource when a customer arrives, she can join the queue and wait. Each customer may balk with a certain probability if there is no available resource and the waiting queue loses customers following a Poisson process with a specific rate under certain conditions. Upon completion of a service, the corresponding unit of resource is released and can be directly used to satisfy a new customer. The arriving, service, balking and reneging rates all depend on the posted price, which are unknown to the decision-maker at the beginning of the planning horizon. Note that under a specific offered price, upon completion of existing services under the previously posted price, the future service process can be reduced to an M/M/c queue (with possibly balking and reneging). The objective is to learn the mappings between rates and prices from past sales and offer the price dynamically to maximize the total expected revenue.

4.1.1 Main Results and Contributions

We propose three new multi-armed bandit (MAB) based learning algorithms for service systems with reusable resources. The performance measure uses cumulative regret, which is the difference between the revenue attained by our learning algorithms and by a clairvoyant optimal pricing policy under full distributional information. When the set of prices is discrete, we propose two algorithms, termed Batch Upper Confidence Bound (BUCB) and Batch Thompson Sampling (BTS), and prove in Theorems 4.4.1 and 4.4.2 that the cumulative regret is $O(\sqrt{PT\log(T)})$, where T is the length of the planning horizon and P is the cardinality of the feasible price set. When the set of prices is continuous with linear dependence on a d_f-dimensional feature vector, we propose a Batch Linear Confidence Bound (BLinUCB) algorithm, and prove in Theorem 4.5.1 that the cumulative regret is $O(d_f \log(T) \log\log(T)\sqrt{T})$. These results match the lower bound up to a logarithmic factor. Our numerical experiments demonstrate and confirm the efficacy of the proposed algorithms.

On a high level, our proposed algorithms contain a Warm-up Phase to eliminate the heavy-tail

effects and a Learning Phase to identify the optimal price. In the Learning Phase, we divide the planning horizon into multiple batches of time periods and iterate the prices over batches. During each batch, we collect the observed demand and service information. At the beginning of each batch, we compute a revenue rate index for each price based on the past observed information (either via computing the upper confidence bound or sampling from the posterior distribution) and choose the price with the highest index for the next batch. One key difference (from the state-of-the-art literature in revenue management with perishable resources, e.g., Ferreira et al. 2018, Chen and Shi 2019b) is that for each batch, we need to further divide it into two intervals. The first interval is for completing serving the existing customers (under previously posted prices) and reaching the steady state of the corresponding M/M/c queue under the newly posted price. The system maintains the steady state of the queue in the second interval.

Compared with the prior pricing and multi-armed/linear bandit literature, there are several salient challenges when solving this problem.

(a) First, the observed information, i.e., the interarrival time and service time of an M/M/c system, are exponentially distributed, thereby inducing ***unbounded and heavy-tailed random rewards***. This is in stark contrast with the sub-Gaussian rewards generally required in the MAB literature, especially in the linear bandit literature. The key is to achieve similar order concentration bounds as for light-tailed sub-Gaussian observations. When the set of prices is discrete, our BUCB and BTS accommodate a special structure to allow a Warm-up Phase to perform a uniform information collection over all candidate prices. This Warm-up Phase invokes Bernstein's inequality and eliminates the negative effects of heavy-tailed sub-exponential random variables (see Proposition 4.3.1). When the set of prices is continuous, to tackle the difficulty of sub-exponential observations of linear bandits, our BLinUCB accumulates a carefully designed set of observations and uses the empirical statistics as data points to update the estimations of regression coefficients with a probabilistic concentration guarantee (see Proposition 4.5.2), which gives the first concentration result for linear bandit with sub-exponential rewards.

(b) Second, a major challenge in the regret analysis lies in bounding the ***loss of nonstationarity*** due to transient system performance. Indeed, as soon as the posted price changes, both the arrival and departure processes change. Moreover, the system may serve customers who arrived during previous time periods with the old price and new customers who arrived in the current time period with the new price at the same time. Therefore, it is difficult to analyze the transient system performance and keep track of the cumulative revenue in a dynamic pricing scheme. To deal with this challenge, we allow the system to take sufficient time to reach the steady state under each set of system parameters before changing the price. We analyze the length of this transient period by developing a new coupling analysis for M/M/c queues upon

price changes (see Propositions 4.4.1 and 4.4.2). To the best of our knowledge, this is the first result giving a finite-time high probability bound on mixing times of M/M/c queues.

(c) Third, under any posted price, the firm may experience the ***loss of customers*** due to lack of available resources, i.e., balking and reneging may happen. Therefore, we need to bound this type of loss, which results in several technical challenges.

(d) Fourth, we derive a simple linear program (LP) for computing the optimal state-independent pricing decisions. We show that the optimal objective value of this LP is an upper bound of the maximum cumulative revenue under full information scenario. We further derive relaxed regret as our performance measure, which is an upper bound of the regret for any pricing policies. We decompose the relaxed regret into two parts. The first part represents the revenue loss induced by the switching of prices, which is discussed in the second challenge. The second part represents the ***loss of suboptimality***, which is induced by not offering the optimal state-independent price in certain time periods. This decomposition also stresses the trade-off between exploration and exploitation. Too much exploration on different prices would increase the regret of nonstationarity, and too little exploration would increase the regret of suboptimality. Our batched MAB approach finds the right and delicate balance.

4.1.2 Literature Review

Our work is closely related to following streams of literature.

4.1.2.1 Revenue Management with Reusable Resources.

The majority of the prior studies focused on revenue management with perishable resources, and we refer interested readers to survey articles by Talluri and Van Ryzin (2006) and Özer et al. (2012) and den Boer (2015) for an overview. Since the literature on revenue management with perishable resources (with and without demand learning) is vast, we refrain ourselves from reviewing a prolonged list but only focus on the literature of revenue management with reusable resources in this section.

Revenue management with reusable resources considers selling (or renting) reusable resources, which requires the firm to properly balance resource utilization across different service cycles. As noted before, one key challenge in the reusable resource setting is the need to dynamically match demands with the varying inventory of reusable resources. Several recent studies focused on admission control in a stationary infinite horizon and single resource setting with a general service time, and the authors developed provably near-optimal heuristic controls with constant factor performance guarantees (see Levi and Radovanović 2010, Chen and Shi 2017, Chen et al. 2017).

Besides admission control, there has been a stream of literature considering static and dynamic pricing. Maglaras (2006) studied static pricing (via a fluid model analysis) in a single reusable resource setting with multiple classes of customers. Araman and Caldentey (2009) formulated the problem as a Poisson intensity control problem and derived structural properties of an optimal solution, and suggested an efficient approximated solution. Xu and Li (2013) studied optimal pricing control under the assumption of exponential service time for cloud computing service providers, and analyzed some properties of the optimal pricing control and the maximum cumulative revenue. Also for cloud computing service pricing, Doan et al. (2020) considered a multi-resource setting under ambiguous distributions of demand and service time, and constructed efficient heuristic fixed-price pricing policies through robust deterministic approximation models. Lei and Jasin (2020) also considered a multi-resource setting but under a deterministic service time assumption, and developed an asymptotically near-optimal real-time pricing control algorithm and further showed that their results can be extended to a more general setting with heterogeneous service time and advance reservation. Besbes et al. (2020) proved that a static pricing policy simultaneously guarantees 78.9% of the profit, market share, and service level from the optimal policy. Owen and Simchi-Levi (2018) considered a joint assortment and pricing problem under a multi-resource setting and provided constant factor performance guarantees for their proposed controls assuming discrete price points and exponential service time. Rusmevichientong et al. (2020) considered a dynamic assortment and pricing problem with reusable products and proposed a provably near-optimal policy based on approximate dynamic programming, which is guaranteed to obtain at least half of the optimal revenue.

The aforementioned literature assumed that the distributional information of the underlying model is known to the decision-maker *a priori*. There has been very little work considering online learning for service systems with reusable resources. The closest work to ours in this space is by Chen et al. (2020b), which was developed concurrently and independently with our work. The authors proposed an online learning framework for a dynamic pricing and capacity sizing problem in a GI/GI/1 queue to maximize the cumulative profit under incomplete information. The problem settings of their work and ours are different in the following aspects: (i) their work assumes that the firm charges a uniform price from each customer while ours assumes that the customers are charged at the product of the unit-time price and the service time; (ii) their work considers holding cost for customers who are waiting in the queue while ours includes no holding cost but allows for balking and reneging; (iii) their work treats the capacity of the resource, i.e., the distribution of the service time, as a decision while our work assumes the service time distribution is also uncertain and affected by the unit price; and (iv) their work considers a GI/GI/1 queue and we study an M/M/c queue. The specific queueing system dynamics are important to the regret analysis of both their work and our paper. In addition to the problem settings, the learning and decision schemes are

also very different: their work develops a stochastic gradient descent (SGD) algorithm to iterate the price and capacity decisions to approach the optimal decision, while we develop a batched multi-armed bandit approach. Despite the above differences, both their work and ours use cumulative regret as the performance measure and both analyses follow a similar structure: firstly quantify the loss of nonstationarity brought from the transient system dynamics of the queueing model and secondly quantify the loss of suboptimality by choosing suboptimal decisions during the learning procedure. However, the detailed analyses of these two types of losses are very different due to differences in the problem settings and and the inherent learning mechanisms. There have been other recent studies in queueing network controls adopting more general reinforcement learning (RL) approaches, such as Dai and Gluzman (2017), Liu et al. (2019), Shah et al. (2020).

4.1.2.2 Multi-Armed Bandit (MAB) and Linear Bandit.

MAB. The MAB literature predominantly assumes that rewards follow sub-Gaussian (including bounded) distributions and proves tight regret bounds based on concentration inequalities of the light-tailed sub-Gaussian distributions (see, e.g., Ledoux 2001, Boucheron et al. 2013). Bubeck et al. (2013) was the first to thoroughly study MAB with heavy-tailed reward distributions. They considered rewards with moments of order $1 + \epsilon$ for some $\epsilon \in (0, 1]$. Their general approach is to replace the empirical mean in the upper confidence index with other robust estimators, such as the truncated empirical mean, Catoni's M-estimator, and the median-of-means estimator. Their results also imply that moments of order 2 (i.e., finite variance) are sufficient to obtain regret bounds of the same order as under sub-Gaussian reward distributions. Besides Bubeck et al. (2013), there have been other studies devoted to MAB with heavy-tailed reward distributions, for example, Vakili et al. (2013) proposed sequential phases of exploration and exploitation; Korda et al. (2013) and Dubey and Pentland (2019) modified Thompson Sampling (TS) algorithms for heavy-tailed rewards; and Xia et al. (2016) studied heavy-tailed rewards under the multi-play setting and Dubey et al. (2020) considered multi-agent setting. A recent work by Jia et al. (2021) developed three UCB algorithms to solve MAB with sub-exponential rewards. Our BUCB and BTS are following the same framework as that of the UCB-Warm in Jia et al. (2021) and our problem is more challenging because our concentration inequality includes two types of observations, i.e., both arrival and service processes.

Linear Bandit. There have been several well-developed studies on linear bandit problems. For example, Li et al. (2010) developed a computationally efficient UCB algorithm for contextual bandits with the help of ridge regression. Abbasi-Yadkori et al. (2011) further improved the regret bound. Filippi et al. (2010) extended the results of linear bandits to a special class of non-linear bandits through Generalized Linear Model (GLM) UCB algorithm, Zhou et al. (2019) consid-

ered delayed reward observations, and Han et al. (2020) studied sequential batch learning. All these general linear bandit algorithms assume sub-Gaussian errors (see, e.g., Assumption 1 in Rusmevichientong and Tsitsiklis 2010), which allow these algorithms to be directly integrated with linear (ridge) regression techniques. However, there has been no existing literature on linear bandit problems with sub-exponential errors. To the best of our knowledge, our regret analysis provides the first theoretical study of linear bandits with sub-exponential observations (which can also be reviewed as sub-exponential rewards) and contributes to the broad landscape of linear bandits literature by delineating the impact of heavy-tailed random rewards.

4.1.2.3 MAB with Infrequent Action Changes.

MAB problems with infrequent action changes have also been studied in recent years, motivated by the fact that many real-world constraints can prevent the decision-maker from conducting extensive experimentation and making frequent action changes. Specifically, Cheung et al. (2017), Chen and Chao (2019), Chen et al. (2020a) discussed practical reasons of infrequent price changes in revenue management and developed pricing algorithms with limited price switches. Before a systematic analysis of the batched algorithms developed, UCB2 (Auer et al. 2002a) and IMPROVED-UCB (Auer and Ortner 2010) were implicitly M-batch policies with $M = \Theta(\log(T))$. Later, Perchet et al. (2016) studied batched MAB in the two-armed case under a static batch design and Gao et al. (2019) extended the algorithms and results to the general multiple-armed case and adaptively determined batch size. There has also been rich literature focusing on batched MAB problems with switching cost. For example, Cesa-Bianchi et al. (2013) considered uniform switching cost in the objective and Simchi-Levi and Xu (2019) treated the finite switching cost budget as a hard constraint. In our work, we consider a static batch design which is similar to the batch length structure of UCB2 (Auer et al. 2002a) and geometric grid design in Gao et al. (2019) and our algorithms consider $O(\log(T))$ number of batches. The departure from related literature is that we do not consider a strict budget on action changes nor an objective for minimizing the action changes. We have this batched action framework to control the loss of the nonstationarity (resulted from the transient time after each action change).

4.1.2.4 Transient Analysis of Queues.

In the queueing literature, the analysis of M/M/c queues under transient state has a long history. The queue length of the transient M/M/1 queue was firstly studied (see, e.g., Ledermann and Reuter 1954). Although the distribution is explicitly known in terms of modified Bessel functions of the first kind, further studies were conducted to understand how the queue evolves over time. For example, researchers approximated the moments of the queue length (see, e.g., Abate and Whitt

(1987) using exponential and hyper-exponential approximations and Abate and Whitt (1988) using Laplace transforms) and investigated generating functions for the partial differential equations (Bailey 1954). The transient analysis techniques were further extended to M/M/c queues. Kelton and Law (1985) evaluated the transient distribution function of the delay in queue of the nth arriving customer based on the analysis of M/M/1 queue in Morisaku (1976). Parthasarathy and Sharafali (1989) derived a time-dependent solution for the queue length based on partial differential equations. Kelton and Law (1985) pointed out that one needs to allow the queue to run "long-enough" for the effect of the initial starting point to have dissipated and collects observations during the ensuing "steady-state" portion of the run and they investigated different starting points to promote rapid convergence. To the best of our knowledge, our result gives the first finite-time high probability bound on mixing times of M/M/c queues (upon action changes). Specifically, we develop a new coupling argument between this target system and a virtual system starting from a state sampled from the steady-state distribution and maintaining the steady state thereafter.

4.1.3 Structure of the Paper

The remainder of this paper is organized as follows. In §4.2, we first describe our problem with a discrete price set and define the regret and the relaxed regret. In §4.3, we describe the BUCB and BTS algorithms. In §4.4, we analyze the regret bound of BUCB and BTS algorithms, where we first provide in §4.4.1 a coupling analysis as an intermediate result of algorithmic performance analysis. In §4.5, we consider the setting with a continuous price set, and propose the BLinUCB algorithm and analyze the regret bound. In §4.6, we extend our results (with discrete price set) to settings where customers may balk or renege when the system does not have enough capacity. In §4.7, we test the numerical performance of our algorithms and benchmark algorithms using diverse instances. In §4.8, we conclude the paper and point out several future research directions.

4.2 Problem Formulation

We consider a service-providing firm who serves customers with reusable resources of finite capacity c over a finite horizon T. At the beginning period $t \leq T$, the firm posts a price p_t from a given candidate set $\mathcal{P}$ to maximize the total cumulative revenue. For now, we assume that $\mathcal{P}$ is finite and denote its cardinality by $P = |\mathcal{P}|$. We shall discuss the model with continuous $\mathcal{P}$ in §4.5. Under the posted price p, $\forall p \in \mathcal{P}$, customers arrive at the system according to a Poisson process with rate λ_p and they are served on a first-arrive-first-serve basis by occupying one unit of the resource with the service time following an exponential distribution with rate μ_p. Customers pay p for per unit time of service. If there is not enough resource capacity, the customer will join

the queue until being serviced. Note that the service system under any posted price $p \in \mathcal{P}$ can be reduced to an M/M/c queue model (when the system only contains customers arriving under this price, i.e., after finishing serving other customers arriving under the previously posted prices). The assumption of exponential service times is standard in the literature (see, e.g., Savin et al. 2005, Gans and Savin 2007, Owen and Simchi-Levi 2018).

We assume that the firm does not know the underlying arrival rate λ_p and service rate μ_p under price p, $\forall p \in \mathcal{P}$, *a priori*. The firm aims at finding a periodic review pricing policy as $\pi : \{(n,t) : n = 0, 1, \ldots, \infty,\ t = 1, \ldots, T\} \to \mathcal{P}$, where the firm selects price $\pi(n,t) \in \mathcal{P}$ for period t when there are n customers in the system at the beginning of period t. The goal is to maximize the expected total revenue. The expected revenue under pricing policy π during period t, $\forall t = 1, \ldots, T$, is denoted by J_t^π and the cumulative expected revenue over periods $\{1, \ldots, T\}$ is denoted by $J^\pi = \sum_{t=1}^{T} J_t^\pi$.

Assumption 4.2.1. *We make the following assumptions on stability, time, and maximum revenue.*

1. *The utilization factor* $\rho_p = \frac{\lambda_p}{c\mu_p} < 1$ *for any candidate price* $p \in \mathcal{P}$.
2. $\log\log(T) \geq 3$.
3. *A constant* $r_{\max}$ *is known such that* $r_{\max} \geq \max_{p \in \mathcal{P}} \lambda_p / \mu_p$.

Assumption 4.2.1 says the following: 1) the system is stable for any candidate price; 2) we require the planning horizon to be sufficiently long, since it takes time for the underlying service system to reach a steady state under any posted price; 3) a valid upper bound of the stationary revenue rate of all candidate prices is known.

Assumption 4.2.2. *For any two candidate prices* $p_h,\ p_l \in \mathcal{P}$ *and* $p_h > p_l$*, we have:*

1. $\lambda_{p_l} \geq \lambda_{p_h}$.
2. $\lambda_{p_l} - \lambda_{p_h} \leq \frac{\rho_l}{-3e \log \rho_l}$.
3. *if* $\mu_{p_l} > \mu_{p_h}$*, then* $\rho_l \geq \rho_h^2$.

Assumption 4.2.2 says the following: 1) the arrival rate under a lower price is higher than or equal to that under a higher price; and 2) the difference in arrival rates between two prices is bounded (by a large constant); and 3) if the service rate under a lower price is higher than that under a higher price, the utilization factor in the former case must be greater than the latter squared. This condition (which is used to justify inequalities (B.10g) and (B.16c)) is rather mild in the following sense: if $\mu_{p_l} \leq \mu_{p_h}$, we do not need any conditions; otherwise, if $\mu_{p_l} > \mu_{p_h}$, a simple sufficient condition to guarantee $\rho_l \geq \rho_h^2$ is $\rho_l \geq \rho_h$, which reduces to $\lambda_{p_l}/\lambda_{p_h} \geq \mu_{p_l}/\mu_{p_h}$, meaning that the ratio of change in arrival rates (with respect to price) exceeds the ratio of change in service rates.

4.2.1 Regret, Relaxed Regret, and Static Price Benchmark

The notion cumulative regret or simply regret is commonly used in online learning (see, e.g., Shalev-Shwartz et al. 2011) to evaluate the performance of a policy if the decision-maker has limited information of the system against the optimal performance under full information. In our problem, the full information means that the firm knows the underlying arrival rate λ_p and service rate μ_p for price p, $\forall p \in \mathcal{P}$. Under full information, we have state-dependent optimal policy $\pi^* = \text{argmax}_\pi J^\pi$ and state-dependent optimal expected revenue $J^* = J^{\pi^*}$. Therefore, we can define the regret of any pricing policy π as the difference between the total state-dependent optimal expected revenue J^* and the total expected revenue under the given policy π, i.e., J^π. In short, we aim at finding heuristic pricing policies that lead to a small regret.

Definition 4.2.1. *The cumulative regret by the end of period T of policy π is defined as:*

$$\textit{Regret}(\pi, T) = J^* - J^\pi. \tag{4.1}$$

However, finding the state-dependent optimal pricing policy π^* requires the firm to solve a dynamic program with an infinite number of potential states, in the form of (n, t), under uncertain parameters, which is computationally intractable. Moreover, the transient system performance is complex and hard to analyze when the offered price changes over periods. Thus, it is impractical or intractable to solve this dynamic program, and further impossible to obtain π^* and J^*. To tackle this problem, we develop an upper bound on the expected revenue achievable by any policy, and therefore, also an upper bound of the state-dependent optimal expected revenue.

Consider continuous decision variables π^p_{nt} for $n = 0, \ldots, \infty,\ t = 1, \ldots, T,\ p \in \mathcal{P}$, a_{nt} for $n = 0, \ldots, \infty,\ t = 1, \ldots, T$, and J_{nt} for $n = 0, \ldots, \infty,\ t = 1, \ldots, T$. We can formulate a linear program (LP) as follows.

$$J^{\text{LP}} = \max_{\pi, a, J} \sum_{n=0}^{\infty} \sum_{t=1}^{T} a_{nt} J_{nt} \tag{4.2a}$$

$$\text{s.t.} \sum_{n=0}^{\infty} a_{nt} = 1 \quad \forall t = 1, \ldots, T \tag{4.2b}$$

$$\sum_{p \in \mathcal{P}} \pi^p_{nt} = 1 \quad \forall n = 0, \ldots, \infty,\ \forall t = 1, \ldots, T \tag{4.2c}$$

$$J_{nt} \leq \sum_{p \in \mathcal{P}} \pi^p_{nt} \frac{\lambda_p}{\mu_p} p \quad \forall n = 0, \ldots, \infty,\ \forall t = 1, \ldots, T \tag{4.2d}$$

$$0 \leq a_{nt} \leq 1 \quad \forall n = 0, \ldots, \infty,\ \forall t = 1, \ldots, T \tag{4.2e}$$

$$0 \leq \pi^p_{nt} \leq 1 \quad \forall n = 0, \ldots, \infty,\ \forall t = 1, \ldots, T,\ \forall p \in \mathcal{P}. \tag{4.2f}$$

Proposition 4.2.1. *The LP solution provides an upper bound on the optimal revenue, i.e.,* $J^* \leq J^{LP}$.

Proof of Proposition 4.2.1. We show that the decisions variables (π, a, J) associated with any admissible policy must satisfy the constraints of the linear program (4.2), and the expected revenue of this admissible policy is exactly the corresponding objective value in (4.2).

Consider an arbitrary policy $\hat{\pi}$. We obtain the values of decision variables by the following steps. Variable $\hat{a}_{nt}$ takes the value of probability that the system has n customers at the beginning of period t under the policy $\hat{\pi}$ for $n = 0, \ldots, \infty$ and $t = 1, \ldots, T$. Therefore, constraints (4.2b) are immediately satisfied. Variable $\hat{\pi}^p_{nt} = 1$ if the policy $\hat{\pi}$ chooses price p at state (n, t), otherwise, $\hat{\pi}^p_{nt} = 0$, for $n = 0, \ldots, \infty$, $t = 1, \ldots, T$, $p \in \mathcal{P}$. As a result, constraints (4.2c) hold because in the left-hand side we have one variable equal to 1 and others all zero. Variable $\hat{J}_{nt}$ takes the value of expected revenue collected before the end of period T from the customers who arrive during period t, when the system has n customers at the beginning of period t, for $n = 0, \ldots, \infty$, $t = 1, \ldots, T$. Consider arbitrary n and t, we denote the price chosen at state (n, t) by the policy $\hat{\pi}$ as $\hat{p}$. Then, by plugging in $\hat{\pi}^{\hat{p}}_{nt} = 1$, in the right-hand side of constraints (4.2d), we have $\frac{\lambda_{\hat{p}}}{\mu_{\hat{p}}}\hat{p}$, where $\lambda_{\hat{p}}$ is the average number of arrived customers during period t, $\frac{1}{\mu_{\hat{p}}}$ is the average service time for each customer, and $\hat{p}$ is the revenue when one customer occupies one unit of resource for one unit of time. Therefore, $\frac{\lambda_{\hat{p}}}{\mu_{\hat{p}}}\hat{p}$ denotes the expected revenue collected from customers who arrive during period t (where assuming all arrived customers being served successfully). Consequently, we can conclude that the right-hand side of (4.2d) is as large as the left-hand side, because the right-hand side may also compute the revenue collected after the end of period T. Hence, constraints (4.2d) are also satisfied. By the value assignment steps of $\hat{a}_{nt}$ and $\hat{J}_{nt}$, for $n = 0, \ldots, \infty$, $t = 1, \ldots, T$, one has that the corresponding LP objective value at solution $(\hat{\pi}, \hat{a}, \hat{J})$ is $J^{\hat{\pi}}$. Therefore, we have $J^{LP} \geq J^{\pi}$, for any admissible policy π, and thus also for the optimal state-dependent policy π^*. **Q.E.D.**

By defining $\tilde{p} = \text{argmax}_{p \in \mathcal{P}} \frac{\lambda_p}{\mu_p} p$, we find an optimal solution $(\tilde{\pi}, \tilde{a}, \tilde{J})$ to the LP in (4.2) as follows.

$$\tilde{\pi}^{\tilde{p}}_{nt} = 1, \quad \forall n = 0, \ldots, \infty, \ \forall t = 1, \ldots, T \tag{4.3a}$$

$$\tilde{\pi}^{p}_{nt} = 0, \quad \forall n = 0, \ldots, \infty, \ \forall t = 1, \ldots, T, \ \forall p \in \mathcal{P} \backslash \{\tilde{p}\} \tag{4.3b}$$

$$\tilde{J}_{0t} = \frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p} \quad \forall t = 1, \ldots, T \tag{4.3c}$$

$$\tilde{J}_{nt} = 0 \quad \forall n = 1, \ldots, \infty, \ \forall t = 1, \ldots, T \tag{4.3d}$$

$$\tilde{a}_{0t} = 1 \quad \forall t = 1, \ldots, T \tag{4.3e}$$

$$\tilde{a}_{nt} = 0 \quad \forall n = 1, \ldots, \infty, \ \forall t = 1, \ldots, T. \tag{4.3f}$$

Then, we can compute

$$J^{\text{LP}} = T\frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p}. \tag{4.4}$$

Note that $\frac{\lambda_p}{\mu_p}p$ is the stationary revenue rate of price p, denoted as $r(\lambda_p, \mu_p, p)$. Thus, we call the price $\tilde{p}$ as the (static) optimal state-independent price, i.e., the price with the highest stationary revenue rate. This result can be seen as the "reusable analog" of the classical static pricing upper bound for revenue management with perishable resources (see Gallego and Van Ryzin 1994).

Corollary 4.2.1. *The single static price policy $\pi^{\tilde{p}} : \pi(n,t) = \tilde{p},\ \forall n = 0, 1, \ldots, \infty,\ \forall t = 1, \ldots, T$ is asymptotically optimal as $T \to \infty$, and we also have $J^{LP} - J^{\pi^{\tilde{p}}} \leq o(\sqrt{T})$.*

Proof of Corollary 4.2.1. This corollary follows directly from Propositions 4.2.1 and 4.4.1 in §4.4.1.1. Based on the above constructed optimal solution $(\tilde{\pi}, \tilde{a}, \tilde{J})$, one can compute that the difference between the revenue upper bound J^{LP} and the collected revenue under the single-price policy $J^{\pi^{\tilde{p}}}$ is from the transient performance of the system when it starts serving customers from an empty state at $t = 1$. The length of the transient state is less than $\log(T)(\log\log(T))^2$ with probability higher than $1 - 2/T^3$ by Proposition 4.4.1. Thus, one can derive that when $T \to \infty$, $J^{\text{LP}} - J^{\pi^{\tilde{p}}} \leq O(\log(T)(\log\log(T))^2)$, which suggests that $\pi^{\tilde{p}}$ is asymptotically optimal. **Q.E.D.**

Definition 4.2.2. *The relaxed regret is defined as*

$$\overline{\textit{Regret}}(\pi, T) = J^{LP} - J^{\pi}. \tag{4.5}$$

Based on Proposition 4.2.1, we have for any policy π, the corresponding relaxed regret is a valid upper bound of its regret, i.e., $\overline{\text{Regret}}(\pi, T) \geq \text{Regret}(\pi, T)$. In the following sections, we demonstrate the performance of proposed algorithms by analyzing the relaxed regret.

Remark 4.2.1. ***The Role of c in the LP Relaxation.*** *There are three aspects. First, a standard Markov chain analysis reveals that for the $M/M/c$ queue in steady-state (under a single price p), the average number of customers in service is precisely $\frac{\lambda_p}{\mu_p}$, which is equal to $c\rho_p$. Thus, the long-run average revenue rate for price p is $\frac{\lambda_p}{\mu_p}p$. It is natural to choose $\tilde{p}$ that maximizes $\frac{\lambda_p}{\mu_p}p$, and the corresponding long-run revenue $J^{LP} = T\frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p}$ (in which c does not show up explicitly). Second, the LP objective J^{π} of any policy π depends on c, because the variable a_{nt} takes the value of probability that the system has n customers at the beginning of period t under this policy π, which depends on c. Note that the LP optimal objective value J^{LP} provides a uniform bound to J^{π} for all admissible policies π. Third, the capacity c and the posted price p attract certain groups of customers, and the arrival and service rates are two attributes of those groups. Thus, the arrival and service rates depend on c and so does J^{LP}. In fact, one could write $J^{LP} = T\frac{\lambda_{\tilde{p},c}}{\mu_{\tilde{p},c}}\tilde{p}$. Once the capacity c changes, the composition of the arrived customers under price p changes and so do the arrival and service rates. Since c is a fixed parameter in our system, we omit the subscript of c in*

$\lambda_{p,c}$ *and* $\mu_{p,c}$.

Remark 4.2.2. ***Continuous versus Discrete Time Horizon.*** *Consider a counterpart model where the firm can continuously change the price at any time between* $[0, T]$*. The optimal LP objective value* J^{LP} *remains a valid upper bound of the revenue collected by the optimal policy even in the continuous time setting. Thus, for any continuous time learning policy* $\tilde{\pi}$*,*

$$\textit{Regret}(\tilde{\pi}, T) = J^* - J^{\tilde{\pi}} \leq J^{LP} - J^{\tilde{\pi}} = \overline{\textit{Regret}}(\tilde{\pi}, T). \tag{4.6}$$

Also, note that any feasible discrete time learning policy clearly belongs to the set of continuous time learning policies. Hence, if we could identify a discrete time learning policy π *that attains the optimal scaling with* T*, then it would be sufficient to work with the discrete time setting. Our numerical result in §4.7.3 also suggests that if a single review period is reasonably small, the revenue flattens out even if we keep reducing its length.*

4.3 Online Learning and Pricing Algorithms: BUCB and BTS

We begin with a brief review of the Upper Confidence Bound algorithm (UCB) and Thompson Sampling algorithm (TS), which have been shown to be effective in solving MAB problems. Then, we propose two new algorithms, namely, the Online Batch Upper Confidence Bound Algorithm (BUCB) in §4.3.2 and the Online Batch Thompson Sampling (BTS) algorithm in §4.3.3.

4.3.1 Preliminaries of UCB and TS

Consider in MAB problems, the player aims to find the arm resulting in the highest expected reward. The stochastic MAB problems consider i.i.d. rewards and the players have no prior information about uncertain rewards. Motivated by adaptive exploration, the concepts of confidence bounds firstly appeared in Lai and Robbins (1985) and the first version of UCB algorithm was proposed in Lai (1987). The UCB algorithm defines an upper confidence bound of the expected reward of each arm based on the past observations, where the actual expected reward is below this bound with a high probability. Therefore, this upper confidence bound can be viewed as an optimistic estimate. In a simple UCB, i.e., UCB1 in Auer et al. (2002b) and Slivkins et al. (2019), it chooses the arm with the highest upper confidence bound, i.e., the best arm based on the optimistic estimate, in each round. There are also many UCB variants (see, e.g., Garivier and Moulines 2011), and our BUCB is based on UCB1 in Auer et al. (2002b) and Slivkins et al. (2019). The UCB algorithms achieve better performance compared with non-adaptive exploration based algorithms. We refer to Chapter 7 in Lattimore and Szepesvári (2020) and Chapter 3 in Slivkins et al. (2019) for more technical details and concrete summaries of UCB algorithms.

As a randomized Bayesian algorithm, TS was firstly proposed by Thompson (1933, 1935). Consider in MAB problems, the player also aims to find the arm resulting in the highest expected reward. The basic idea of TS is that, at each round, random samples are generated by the posterior distribution of the reward for each arm, and then the arm with the highest sampled reward is chosen. Different from the stochastic bandits, the player holds other information about the reward associated with each arm, i.e., the problem instance is drawn from a known prior distribution. Therefore, the remarkable difference of TS from UCB lies in that TS holds priors for the distributions of rewards. The MAB problems under this Bayesian assumption is referred to as Bayesian MAB (see, e.g., Chapter 3 in Slivkins et al. 2019). TS is also known as the "probability matching" property because the probability of an arm being chosen matches the posterior probability that this arm has the highest expected reward. TS gradually becomes a representative algorithm for solving Bayesian MAB and it has been adapted to various Bayesian bandit problem settings (Russo and Van Roy 2014). Compared to other popular MAB algorithms, TS achieves similar theoretical performance guarantees and often better empirical performance (see Kaufmann et al. 2012, Chapelle and Li 2011). We refer interested readers to Russo et al. (2018) and Agrawal and Goyal (2013) and two books by Slivkins et al. (2019) and Lattimore and Szepesvári (2020) for technical details of TS. A similar work to our paper is by Ferreira et al. (2018), where the authors considered a price-based network revenue management problem (with perishable resources) and proposed a pricing algorithm based on TS, to address the challenge of balancing the exploration-exploitation trade-off.

4.3.2 Online Batch Upper Confidence Bound Algorithm (BUCB)

We leverage the idea of UCB to propose an Online Batch Upper Confidence Bound Algorithm (BUCB) for our problem setting described in §4.2. The UCB algorithm cannot be directly used in our setting because we cannot immediately gain or estimate the revenue that we shall collect after offering a price. One of the major difficulties of this problem is to quantify the loss of nonstationarity due to the transient performance of the service system when the price changes.

We separate T periods into two phases. The first phase is the *Warm-up* Phase, where we offer each price one by one and collect the system dynamic information. Once we collect $8\log(T)$ number of service time observations under the current price, we change to the next price in next period. Therefore, the total length of this Warm-up Phase is $O(P\log(T))$. This Warm-up Phase is specifically designed for the exponential observations of M/M/c queues. After this Warm-up Phase, the algorithm can eliminate the sub-exponential tail effect caused by the exponential service time and Poisson arrival process. Then, we can apply a concentration bound as tight as we can do for sub-Gaussian tails. This Warm-up framework and the correspondent concentration bound for

eliminating sub-exponential tails can also be applied to general MAB problems (see Jia et al. 2021). The second phase is *Learning* Phase. We separate the Learning Phase into M number of batches. Define $I_m = 2^m,\ \forall m = 1, \ldots, M$. Each batch m contains $I_m \tau$ periods, where $\tau = \log(T)(\log\log(T))^2$. Note that the batch size is exponentially increasing. By the end of batch m, we have the following observations associated with each $p \in \mathcal{P}$ on hand. We denote $n_m(p)$ as the number of arrived customers under price p. For each customer i arrived under the price p, we denote $\hat{d}_i(p)$ as the observed time interval between the arrival time of customer i and that of customer $i-1$. Then, we compute $\breve{d}_m(p) = \frac{n_m(p)}{\sum_{i=1}^{n_m(p)} \hat{d}_i(p)}$ as an empirical estimate of the arrival rate. Let $n_m^s(p)$ represent the number of customers who have been successfully serviced. For these customers, we denote the observed service time of customer i by $\hat{g}_i(p)$ and compute the empirical mean of the service time as $\bar{g}_m(p) = \frac{\sum_{i=1}^{n_m^s(p)} \hat{g}_i(p)}{n_m^s(p)}$. Therefore, we have that $n_m(p) \propto n_m^s(p),\ \forall m = 1, \ldots, M,\ p \in \mathcal{P}$. Then, one has the following concentration inequality.

Proposition 4.3.1. *For any price $p \in \mathcal{P}$ by the end of batch m, $\forall m = 1, \ldots, M$, if the number of service time observations $n_m^s \geq 8\log(T)$, then we have*

$$\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \geq 1 - \frac{4}{T^4},$$

where $r_{\max}$ is an upper bound of the stationary revenue rate of all candidate prices, i.e., $r_{\max} \geq \max_{p\in\mathcal{P}} \lambda_p/\mu_p$ (see Assumption 4.2.1).

The proof of Proposition 4.3.1 is based on the concentration inequality for sub-exponential random variables (see, e.g., Boucheron et al. 2013) and is provided in Appendix B.2. This result shows that after observing a certain amount of observations, the negative effects of the heavy-tailed distribution has been eliminated and the concentration inequality is as tight as that for sub-Gaussian random variables. Hence, we define the corresponding upper and lower confidence bounds of the stationary revenue rate associated with price p by the end of batch m as follows.

Definition 4.3.1. *The upper confidence bound of the revenue rate associated with price p by the end of batch m is:*

$$U_m(p) = \left(\bar{g}_m(p)\breve{d}_m(p) + 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)p, \tag{4.7}$$

and the lower confidence bound of the revenue rate of price p by the end of batch m is:

$$L_m(p) = \left(\bar{g}_m(p)\breve{d}_m(p) - 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)p. \tag{4.8}$$

Define $\mathbf{Rad}_m(p) = 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}p$ as the confidence radius of price p by end of batch m. Then, for some $\gamma > 0$, two properties hold (see, e.g., Slivkins et al. 2019):

$$\mathbb{E}[(r(\lambda_p, \mu_p, p) - U_m(p))^+] \leq \frac{\gamma}{PT}, \tag{4.9}$$

$$\mathbb{E}[(L_m(p) - r(\lambda_p, \mu_p, p))^+] \leq \frac{\gamma}{PT}. \tag{4.10}$$

At the beginning of each batch $m+1$, BUCB chooses price p with the highest $U_m(p)$ and offers this price during batch $m+1$. We present the algorithmic details in Algorithm 4.1. When offering price p, it is possible that one current customer in service was accepted under another price p', which was provided in previous periods. Therefore, we record arrival information of price p and also service information of any other price $p' \in \mathcal{P}$.

Algorithm 4.1 Online Batch Upper Confidence Bound Algorithm (BUCB).

Input: T, $\mathcal{P}$.
Initialize: τ, I_m, M, $m = 0$.
Warm-up Phase:
for $p \in \mathcal{P}$ **do**
  Offer p, record $\hat{d}_i(p)$ for arriving customers and $\hat{g}_i(p'),\ \forall p' \in \mathcal{P}$ for leaving customers.
  if $n_m^s(p) \geq 8\log(T)$ **then**
    Continue.
  end if
end for
Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $n_m(p)$, $n_m^s(p)$, and $U_m(p)$, $\forall p \in \mathcal{P}$.
Learning Phase:
for $m = 1, \ldots, M$ **do**
  Choose $p_m = \text{argmax}_{p\in\mathcal{P}} U_{m-1}(p)$.
  Offer p_m in batch m, i.e., for $I_m\tau$ periods.
  Record $\hat{d}_i(p_m)$ for arriving customers and $\hat{g}_i(p),\ \forall p \in \mathcal{P}$ for leaving customers.
  Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $n_m(p)$, and $n_m^s(p)$, $\forall p \in \mathcal{P}$.
  Update $U_m(p),\ \forall p \in \mathcal{P}$.
end for

4.3.3 Online Batch Thompson Sampling Algorithm (BTS)

Consider the firm holds prior information about the rates associated with each price $p \in \mathcal{P}$, and thus we can add the Bayesian assumption to this revenue management problem, where the problem instance is drawn from a known prior distribution. By the conjugate prior theory (see, e.g., Raiffa and Schlaifer 1961, Diaconis and Ylvisaker 1979), we assume that BTS holds Gamma priors on λ_p and μ_p, $\forall p \in \mathcal{P}$ and the posteriors are updated by the end of batch m, $\forall m = 1, \ldots, M$.

Specifically, we set the prior distribution of $\lambda_p,\ \forall p \in \mathcal{P}$ as **Gamma**$(\alpha^{\lambda_p}, \beta^{\lambda_p})$. After observing one arrival time interval $\hat{d}_i(p)$ while offering price p, the posterior distribution of λ_p is computed as **Gamma**$(\alpha^{\lambda_p} + 1, \beta^{\lambda_p} + \hat{d}_i(p))$. Similarly, we set the prior distribution of $\mu_p,\ \forall p \in \mathcal{P}$ as **Gamma**$(\alpha^{\mu_p}, \beta^{\mu_p})$ and thus the posterior distribution of μ_p is **Gamma**$(\alpha^{\mu_p} + 1, \beta^{\mu_p} + \hat{g}_i(p))$ after observing one service time $\hat{g}_i(p)$. To implement BTS, at the beginning of batch m, we sample parameters $\hat{\lambda}_p$ and $\hat{\mu}_p$ independently following the updated posterior distributions and then offer the price with the maximum sampled stationary revenue rate $r(\hat{\lambda}_p, \hat{\mu}_p, p) = \frac{\hat{\lambda}_p}{\hat{\mu}_p} p$. We present the algorithmic details in Algorithm 4.2.

Algorithm 4.2 Online Batch Thompson Sampling Algorithm (BTS).

Input: T, $\mathcal{P}$, $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \forall p \in \mathcal{P}$.
Initialize: τ, I_m, M.
Warm-up Phase:
for $p \in \mathcal{P}$ **do**
  Offer p, record $\hat{d}_i(p)$ for arriving customers and $\hat{g}_i(p'),\ \forall p' \in \mathcal{P}$ for leaving customers.
  if $n_m^s(p) \geq 8\log(T)$ **then**
    Continue.
  end if
end for
Update $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \forall p \in \mathcal{P}$.
Learning Phase:
for $m = 1, \ldots, M$ **do**
  Sample $\hat{\lambda}_p \sim$ **Gamma**$(\alpha^{\lambda_p}, \beta^{\lambda_p})$ and $\hat{\mu}_p \sim$ **Gamma**$(\alpha^{\mu_p}, \beta^{\mu_p})$, $\forall p \in \mathcal{P}$.
  Choose $p_m = \text{argmax}_{p \in \mathcal{P}} r(\hat{\lambda}_p, \hat{\mu}_p, p)$.
  Offer p_m in batch m, i.e., for $I_m \tau$ periods.
  Record $\hat{d}_i(p_m)$ for arriving customers and $\hat{g}_i(p),\ \forall p \in \mathcal{P}$ for leaving customers.
  Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $n_m(p)$, and $n_m^s(p)$, $\forall p \in \mathcal{P}$.
  Update $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \forall p \in \mathcal{P}$.
end for

4.4 Performance Analysis of BUCB and BTS

We analyze the performance by the relaxed regret defined in Definition 4.2.2. First, we derive the probability that (i) a system reaches the steady state after a certain amount of time starting from an empty state (i.e., zero customers in the system) under a fixed price (§4.4.1.1) and (ii) a system reaches the steady state after a certain amount of time starting from the steady state under another price (§4.4.1.2). Then, based on the probability analysis, we provide the regret bound for BUCB in §4.4.2 and for BTS in §4.4.3. A high-level roadmap of our analysis is shown in Figure 4.1.

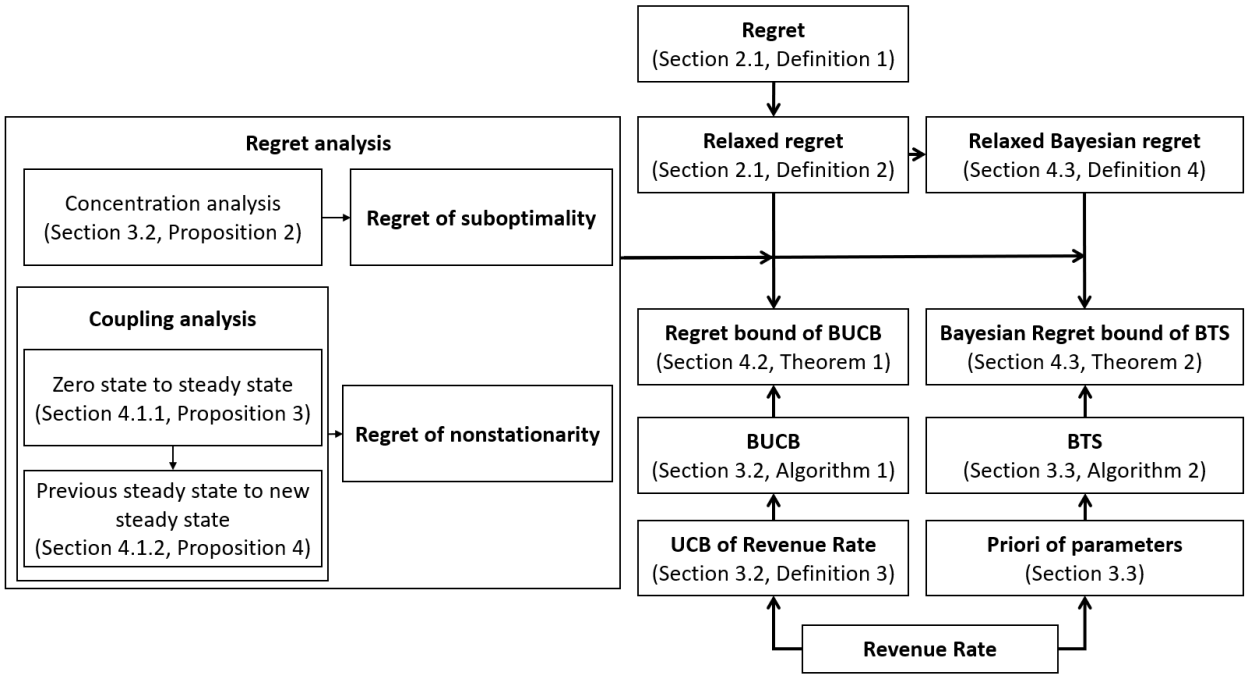


Figure 4.1: Roadmap of BUCB and BTS Algorithms and Regret Analysis.

4.4.1 High Probability Bound for Mixing Time

When the firm starts service from the empty state or switches to another price from the currently offered price, the service (queueing) system enters a transient state and needs a certain amount of time to reach the steady state again. Therefore, the time spent on reaching the steady state, which is also referred to as the mixing time, is crucial for computing the practical revenue under a specific policy and thus crucial for the regret analysis. To the best of our knowledge, this is the first result giving a finite-time high probability bound on mixing times of M/M/c queues (upon action changes). Specifically, we develop a new coupling argument between this target system and a virtual system starting from a state sampled from the steady-state distribution and maintaining the steady state thereafter.

4.4.1.1 Coupling Analysis for M/M/c Queue Starting from Empty State.

Without loss of generality, we focus on a system under fixed arrival rate λ and service rate μ. Let S_t denote the target system that starts from the empty state and $\hat{S}_t$ denote the virtual system that starts with a state sampled from steady-state distribution and remains thereafter. Let S_∞ denote the steady-state overshoot. The random variable S_t $(\hat{S}_t)$, $t \geq 0$ denotes the number of customers in the system at time t of system S_t $(\hat{S}_t)$. The analysis in this section relies on busy period A_n, $\forall n = \{1, \ldots, \infty\}$, defined as the minimum time that the system takes to have $n-1$ customers if starting

with n customers in the system, i.e., $A_n = \min\{t\colon n$ customers in the system at time 0^+ and $n-1$ customers in the system at time $t\}$. Note that $A_n,\ n = 1, \ldots, c$, are random variables following different distributions due to difference in the number of servers in use, while $A_n,\ n = c, \ldots, \infty$, are i.i.d. random variables (Omahen and Marathe 1978, Daley and Servi 1998). The analysis also considers the first-order stochastic dominance between two random variables defined in Definition 1 in Hadar and Russell (1969) and in Seth and Yalonetzky (2014).

Lemma 4.4.1. ***(First-order Stochastic Dominance between A_i and A_j in M/M/c Queue)*** *For random variables A_i and A_j where $i < j \leq c$, A_i first-order dominates A_j, i.e., $\mathbb{P}(A_j \leq x) > \mathbb{P}(A_i \leq x)$ for any $x > 0$.*

To prove Lemma 4.4.1, we construct a virtual queue and use the busy periods of this virtual queue as intermediate values to establish the first-order stochastic dominance between A_i and A_j. The detailed proof is provided in Appendix B.3. For $t = 0, \ldots, T$, define events

$$\mathcal{A}_t = \{S_t = \hat{S}_t \text{ after time } t\}. \tag{4.11}$$

Therefore, systems S_t and $\hat{S}_t$ couple after t is equivalent to that event $\mathcal{A}_t$ happens, $\forall t \geq 0$.

Proposition 4.4.1. ***(Coupling Probability of M/M/c Queue starting from Empty State)*** *For $t \geq \tau$, where $\tau = \log(T)(\log\log(T))^2$, the probability that $\hat{S}_t$ and S_t couple is bounded as follows.*

$$\mathbb{P}(\mathcal{A}_\tau) \geq 1 - \frac{2}{T^3}.$$

The proof of Proposition 4.4.1 is provided in Appendix B.3, which follows the law of total probability, the length of the time horizon T in Assumption 4.2.1, the first-order stochastic dominance in Lemma 4.4.1, and the concentration inequality for independent samples in Lemma B.3.1.

Remark 4.4.1. *Note that this result gives a uniform lower bound for all system parameters; one could obtain a parametric lower bound from the proof (that is decreasing in ρ), which suggests that it is slower to reach stationarity when the utilization factor ρ increases.*

4.4.1.2 Coupling Analysis for M/M/c Queue when Price Changes.

Note that when implementing our learning algorithms, the $M/M/c$ queue does not really start from empty state. This is because whenever a new price is posted, the customers arrived under the previous posted price still remain in the system. We need to ensure that the coupling occurs with high probability even with price changes (with the aid of Assumption 4.2.2), which is encapsulated in Proposition 4.4.2.

Proposition 4.4.2. ***(Coupling Probability of M/M/c Queue when Price Changes)*** *The probability that the system reaches the steady state within 2τ after the price changes is bounded by $1 - \frac{4}{T^3}$,*

where $\tau = \log(T)(\log\log(T))^2$.

The proof of Proposition 4.4.2 is based on Proposition 4.4.1 and Assumption 4.2.2. Specifically, we decompose possible price changes into the following two cases. The first is when the price changes from a lower price p_l to a higher price p_h and the second is vice versa. For each case, we further consider two sub-cases, (i) the unit service rate $\mu_{p_l} > \mu_{p_h}$ and (ii) $\mu_{p_l} \leq \mu_{p_h}$. For each case, we construct a virtual system and derive the coupling probability of the corresponding virtual system. Then we show that the actual coupling probability is bounded from below by the probability with which the virtual system reaches the steady state. The full proof is given in Appendix B.3.

4.4.2 Performance of BUCB

With the high probability bound for mixing time developed in §4.4.1 and the key result in Proposition 4.4.2, we are ready to carry out the regret analysis of BUCB.

Theorem 4.4.1. *The T-period cumulative regret of BUCB is bounded by $O(\sqrt{PT\log(T)})$.*

Note that the regret lower bound for nominal MAB is $\Omega(\sqrt{PT})$ (see, e.g., Theorem 2.1 in Slivkins et al. (2019)), which suggests that our regret bound is tight, up to a logarithmic factor.

Proof of Theorem 4.4.1. We bound the relaxed regret during the Warm-up and Learning Phases separately. We further decompose the relaxed regret during the Learning Phase into two parts. The first part is the regret of suboptimality, which is induced by not offering the state-independent optimal price in certain time periods. The second part is the regret of nonstationarity, which represents the revenue loss induced by changing offered prices.

Regret of the Warm-up Phase. The difference between $J^*_{\text{Warm-up}}$ and $J^{\text{BUCB}}_{\text{Warm-up}}$ is at most linear on the length of the Warm-up Phase, which is $O(P\log(T))$. Therefore we can directly have

$$\overline{\text{Regret}}_{\text{Warm-up}}(\pi_{\text{BUCB}}, T) = O(P\log(T)). \tag{4.12}$$

Regret of nonstationarity during the Learning Phase. We denote the offered price of batch m in BUCB as p_m. Therefore, the expected revenue collected in batch m is

$$J_m(p_m) = J_{m,0\sim 2\tau}(p_m) + J_{m,2\tau\sim I_m\tau}(p_m),$$

where $J_{m,0\sim 2\tau}(p_m)$ denotes the expected revenue collected in the first 2τ periods and $J_{m,2\tau\sim I_m\tau}(p_m)$ denotes the expected revenue collected from $(2\tau+1)$th period to the end of batch m. We define event $\mathcal{C}_{2\tau}$ as that the system reaches the steady state under the new price within the first 2τ periods and define $\bar{\mathcal{C}}_{2\tau}$ as the complementary event of $\mathcal{C}_{2\tau}$. By Proposition 4.4.2, we have $\mathbb{P}[\mathcal{C}_{2\tau}] \geq 1 - \frac{4}{T^3}$. Consequently, we define $J_{m,\mathcal{C}_{2\tau}}(p_m)$ to be the expected value of $J_{m,2\tau\sim I_m\tau}(p_m)$

conditional on $\mathcal{C}_{2\tau}$ and $J_{m,\bar{\mathcal{C}}_{2\tau}}(p_m)$ similarly. As a result, we can express the expected revenue during batch m as:

$$\begin{aligned}
J_m(p_m) &= J_{m,0\sim 2\tau}(p_m) + J_{m,\mathcal{C}_{2\tau}}(p_m)\mathbb{P}(\mathcal{C}_{2\tau}) + J_{m,\bar{\mathcal{C}}_{2\tau}}(p_m)\big(1-\mathbb{P}(\mathcal{C}_{2\tau})\big) \\
&\geq O(\tau) + \left(\frac{p_m\lambda_{p_m}}{\mu_{p_m}}\right)(I_m\tau - 2\tau)\left(1-\frac{4}{T^3}\right) + O\left(\frac{I_m\tau - 2\tau}{T^3}\right) \\
&= \left(\frac{p_m\lambda_{p_m}}{\mu_{p_m}}\right) I_m\tau - O(\tau) \\
&= r(\lambda_{p_m}, \mu_{p_m}, p_m) I_m\tau - O(\tau).
\end{aligned} \tag{4.13}$$

Regret of suboptimality during the Learning Phase. Recall the upper and lower confidence bounds of the revenue rate defined in Definition 4.3.1. It is clear that

$$U_m(p) - L_m(p) = 2\mathbf{Rad}_m(p) = 8\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}p.$$

For batch m, we can compute the difference in the revenue rates between the optimal price and selected price p_m as

$$\Delta(p_m) = r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - r(\lambda_{p_m}, \mu_{p_m}, p_m) \tag{4.14a}$$

$$= \big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(p_m)\big) + \big(U_m(p_m) - L_m(p_m)\big) + \big(L_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big) \tag{4.14b}$$

$$\leq \big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big) + \big(U_m(p_m) - L_m(p_m)\big) + \big(L_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big), \tag{4.14c}$$

where the last inequality holds because $U_m(p_m) \geq U_m(\tilde{p})$ by the algorithm construction.

Then the cumulative regret during the Learning Phase is

$$\overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) = J^{\text{LP}}_{\text{Learning}} - J^{\pi_{\text{BUCB}}}_{\text{Learning}} \tag{4.15a}$$

$$= \sum_{m=1}^{M}\left(r\left(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}\right) I_m\tau - J_m\left(p_m\right)\right) \tag{4.15b}$$

$$= \sum_{m=1}^{M}\mathbb{E}\Big[\Delta(p_m)I_m\tau + O(\tau)\Big]. \tag{4.15c}$$

By (4.14), we have

$$\sum_{m=1}^{M} \mathbb{E}\Big[\Delta(p_m) I_m \tau\Big] = \sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big)\Big] I_m \tau + \sum_{m=1}^{M} \mathbb{E}\Big[\big(U_m(p_m) - L_m(p_m)\big)\Big] I_m \tau + \sum_{m=1}^{M} \mathbb{E}\Big[\big(L_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big)\Big] I_m \tau. \tag{4.16}$$

We first analyze the middle term of (4.16). Let $\mathcal{M}_p$ denote the batches choosing price p and T_p the number of periods choosing price p, i.e., $T_p = \sum_{m \in \mathcal{M}_p} I_m \tau$. Then,

$$\sum_{m=1}^{M} \mathbb{E}\Big[\big(U_m(p_m) - L_m(p_m)\big) I_m \tau\Big] = \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}_p} \mathbb{E}\big[U_m(p) - L_m(p)\big] I_m \tau \tag{4.17a}$$

$$= \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}_p} 2\mathbb{E}\big[\mathbf{Rad}_m(p)\big] I_m \tau \tag{4.17b}$$

$$\leq \sum_{p \in \mathcal{P}} O\left(\sqrt{T_p \log(T)}\right) \tag{4.17c}$$

$$= O\left(\sqrt{\log(T)}\right) \sum_{p \in \mathcal{P}} \sqrt{T_p} \tag{4.17d}$$

$$\leq O\left(\sqrt{PT \log(T)}\right), \tag{4.17e}$$

where (4.17c) by Lemma B.4.1 (see Appendix B.4) and (4.17e) follows Jensen's inequality.

Below we analyze the first term of (4.16), and the same logic applies to the third term, of which

we omit the details here.

$$\sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big)\Big] I_m \tau \tag{4.18a}$$

$$\leq \sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big)\mathbb{1}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big)\Big] I_m \tau \tag{4.18b}$$

$$\leq \sum_{m=1}^{M} \mathbb{E}\Big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\mathbb{1}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big) - U_m(\tilde{p})\mathbb{1}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big)\Big] I_m \tau \tag{4.18c}$$

$$= \sum_{m=1}^{M} r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\mathbb{P}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big) I_m \tau - \sum_{m=1}^{M} \mathbb{E}\Big[U_m(\tilde{p})\mathbb{1}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big)\Big] I_m \tau \tag{4.18d}$$

$$\leq \sum_{m=1}^{M} r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\mathbb{P}\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) > U_m(\tilde{p})\big) I_m \tau \tag{4.18e}$$

$$\leq O\left(\frac{1}{T^3}\right), \tag{4.18f}$$

where (4.18e) is because $U_m(\tilde{p}) > 0$ by Definition 4.3.1, and (4.18f) follows Proposition 4.3.1.

Given that $O\big(n_M(p)\big) = O(T_p)$, and the total number of batches is $\log_2\left(1 + \frac{T - 8P\log(T)}{2\log(T^2)(\log\log(T))^3}\right) = O\big(\log(T)\big)$, we have $\sum_{m=1}^{M} O(\tau) = O\big((\log(T))^2(\log\log(T))^2\big)$. Combining the three terms together,

$$\begin{aligned}\overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) =& \sum_{m=1}^{M} \mathbb{E}\Big[\Delta(p_m) I_m \tau\Big] + \sum_{m=1}^{M} O(\tau) \\ =& O\left(\sqrt{PT\log(T)}\right) + O\left(\frac{1}{T^3}\right) + O\big((\log(T))^2(\log\log(T))^2\big) \\ =& O\left(\sqrt{PT\log(T)}\right).\end{aligned} \tag{4.19}$$

Total Regret of BUCB algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of BUCB algorithm as:

$$\overline{\text{Regret}}(\pi_{\text{BUCB}}, T) = \overline{\text{Regret}}_{\text{Warm-up}}(\pi_{\text{BUCB}}, T) + \overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) \tag{4.20a}$$

$$= O(P\log(T)) + O\left(\sqrt{PT\log(T)}\right). \tag{4.20b}$$

Q.E.D.

Remark 4.4.2. ***Necessity of Batched MAB Approach.*** *For our system, one could first learn the*

arrival rates and service times (thereby revenue rates) for each candidate price $p \in \mathcal{P}$ and then optimize. Whenever a new price is posted, there is complication of waiting for the "old" customers who arrived under the previous posted price to exit the system. However, consider a much simpler fictitious system which could "boot" customers in the queue at will without any penalty costs. Consider the following simple algorithm based on uniform exploration:

(1) Exploration phase: try each price N times and collect the rates; whenever posting a new price, boot any old customers in the queue (who arrived under the previous price).

(2) Select the price $\hat{p}$ with the highest average empirical reward (breaking ties arbitrarily);

(3) Exploitation phase: stick to price $\hat{p}$ in all remaining time $T - PN$.

It is not hard to show that we should set $N = \tilde{O}((T/P)^{2/3})$ inducing a regret of $\tilde{O}(T^{2/3})$, which is not optimal (see, e.g., Slivkins et al. (2019)). With inventory/capacity constraint, Besbes and Zeevi (2009) gave a regret bound $\tilde{O}(T^{3/4})$ for a similar uniform exploration strategy for the revenue management model under perishable resources. Note that our system with reusable resources is conceivably much harder, which spurs the necessity of a batched MAB approach. For each batch, we post a candidate price and evaluate its performance after the system reaches stationarity. This evaluation necessitates a careful finite time analysis on mixing times of the underlying queueing system upon price changes. Not only do we achieve optimal theoretical regret, but we also significantly outperform the explore-then-commit strategies numerically.

4.4.3 Performance of Online Batch Thompson Sampling (BTS)

BTS holds priori distributions for the arrival and service rates, and thus we use the Bayesian regret as our new performance measure, which is defined as the expected regret of the instance in expectation over all problem instances. Similarly, we directly define a relaxed Bayesian regret as a valid upper bound of the Bayesian regret. The relaxed Bayesian regret bound of BTS is in the same order as that we obtain for BUCB approach in §4.4.2.

Definition 4.4.1. *Denoting all the events before period t by $\mathcal{F}_t$, $\forall t \geq 0$, we define the relaxed Bayesian Regret of policy π by period T as*

$$\overline{BR}(\pi, T) = \mathbb{E}_{(\lambda,\mu)\sim priori}\big[\mathbb{E}[J^{LP} - J^{\pi}|\mathcal{F}_T]\big].$$

Theorem 4.4.2. *The T-period cumulative Bayesian regret of BTS is bounded by $O(\sqrt{PT\log(T)})$.*

Proof of Theorem 4.4.2. The analysis here is similar to that in the proof of Theorem 4.4.1. We first bound the Bayesian regret of the Warm-up Phase and then for the Learning Phase.

Bayesian regret in the Warm-up Phase. Similarly, we can bound the Bayesian regret in the Warm-up Phase as

$$\overline{\mathrm{BR}}_{\text{Warm-up}}(\pi_{\text{BTS}}, T) = O(P\log(T)). \tag{4.21}$$

Bayesian regret in the Learning Phase. In batch m, the chosen price p_m and the state-independent optimal price $\tilde{p}$ are identically distributed conditional on $\mathcal{F}_m$, which indicates that $\mathbb{E}\big[U_m(p_m) - U_m(\tilde{p})|\mathcal{F}_m\big] = 0$ (e.g., see the proof of Lemma 3.7 in Slivkins et al. (2019)). Thus, the Bayesian regret in batch m is:

$$\begin{aligned}
\overline{\mathrm{BR}}_m &= \mathop{\mathbb{E}}_{(\lambda,\mu)\sim\text{priori}}\Big[\mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})I_m\tau - J_m(p_m)|\mathcal{F}_m\big]\Big] && \text{(4.22a)}\\
&= \mathop{\mathbb{E}}_{(\lambda,\mu)\sim\text{priori}}\Big[\mathbb{E}\big[I_m\tau U_m(p_m) - J_m(p_m) + r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})I_m\tau - I_m\tau U_m(\tilde{p})|\mathcal{F}_m\big]\Big] && \text{(4.22b)}\\
&= \mathop{\mathbb{E}}_{(\lambda,\mu)\sim\text{priori}}\Big[\mathbb{E}\big[I_m\tau U_m(p_m) - J_m(p_m)|\mathcal{F}_m\big]\Big] + \mathop{\mathbb{E}}_{(\lambda,\mu)\sim\text{priori}}\Big[\mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})I_m\tau - I_m\tau U_m(\tilde{p})|\mathcal{F}_m\big]\Big] && \text{(4.22c)}\\
&= \mathbb{E}\big[I_m\tau U_m(p_m) - J_m(p_m)\big] + \mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})I_m\tau - I_m\tau U_m(\tilde{p})\big] && \text{(4.22d)}\\
&\le \mathbb{E}\big[I_m\tau U_m(p_m) - I_m\tau\cdot r(\lambda_{p_m}, \mu_{p_m}, p_m) + O(\tau)\big] + \mathbb{E}\big[I_m\tau\cdot r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - I_m\tau U_m(\tilde{p})\big] && \text{(4.22e)}\\
&= I_m\tau\Big(\mathbb{E}\big[U_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big] + \mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big]\Big) + O(\tau). && \text{(4.22f)}
\end{aligned}$$

Following two properties of the upper and lower bounds of the stationary revenue rate in (4.9) and (4.10) (derived using Proposition 4.3.1), we have

$$\mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big] \le \mathbb{E}\left[(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p}))^+\right] \le \sum_{p\in\mathcal{P}} \mathbb{E}\left[(r(\lambda_p, \mu_p, p) - U_m(p))^+\right] \le \frac{\gamma}{T}, \tag{4.23}$$

and similarly,

$$\begin{aligned}
\mathbb{E}\big[U_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big] =& \mathbb{E}\big[2\cdot\mathbf{Rad}_m(p_m) + L_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big]\\
\le& \mathbb{E}\big[2\cdot\mathbf{Rad}_m(p_m)\big] + \mathbb{E}\big[\,(L_m(p) - r(\lambda_{p_m}, \mu_{p_m}, p_m))^+\,\big]\\
\le& \mathbb{E}\big[2\cdot\mathbf{Rad}_m(p_m)\big] + \sum_{p\in\mathcal{P}}\mathbb{E}\Big[\,(L_m(p) - r(\lambda_{p_m}, \mu_{p_m}, p_m))^+\,\Big]\\
\le& \mathbb{E}\big[2\cdot\mathbf{Rad}_m(p_m)\big] + \frac{\gamma}{T}.
\end{aligned} \tag{4.24}$$

Summing over all batches $m, \forall m = 1, \ldots, M$, we have the cumulative relaxed Bayesian regret

by the end of the Learning Phase as:

$$\overline{\text{BR}}_{\text{Learning}}(\pi_{\text{BTS}}, T) = \sum_{m=1}^{M} \overline{\text{BR}}_m \tag{4.25a}$$

$$= \sum_{m=1}^{M} \left(I_m \tau \Big(\mathbb{E}\big[U_m(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big] + \mathbb{E}\big[r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_m(\tilde{p})\big] \Big) + O(\tau) \right) \tag{4.25b}$$

$$\leq \sum_{m=1}^{M} \left(I_m \tau \left(2\frac{\gamma}{T} + 2\mathbb{E}\big[\mathbf{Rad}_m(p_m)\big] \right) \right) + O\big(\tau \log(T)\big) \tag{4.25c}$$

$$\leq 2 \sum_{m=1}^{M} I_m \tau \frac{\gamma}{T} + 2 \sum_{p \in \mathcal{P}} \sum_{m \in \mathcal{M}_p} I_m \tau \mathbb{E}\big[\mathbf{Rad}_m(p)\big] + O\big(\tau \log(T)\big) \tag{4.25d}$$

$$= 2\gamma + \sum_{p \in \mathcal{P}} O\left(\sqrt{T_p \log(T)} \right) + O\big(\tau \log(T)\big) \tag{4.25e}$$

$$\leq O\left(\sqrt{PT \log(T)} \right) + O\big((\log(T))^2 (\log\log(T))^2\big) \tag{4.25f}$$

$$= O\left(\sqrt{PT \log(T)} \right), \tag{4.25g}$$

where (4.25e) is by Lemma B.4.1 (see Appendix B.4) and (4.25f) is by Jensen's inequality.

Regret of BTS algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of BTS algorithm as:

$$\overline{\text{BR}}(\pi_{\text{BTS}}, T) = \overline{\text{BR}}_{\text{Warm-up}}(\pi_{\text{BTS}}, T) + \overline{\text{BR}}_{\text{Learning}}(\pi_{\text{BTS}}, T) \tag{4.26a}$$

$$= O(P \log(T)) + O\left(\sqrt{PT \log(T)} \right). \tag{4.26b}$$

Q.E.D.

4.5 Model with Continuous Price Set under Contextual Information

We now consider a continuous price setting where the firm can choose any price between a fixed range $[p_L, p_U]$. We consider the same M/M/c queue as in §4.2 without balking and reneging.

We assume a linear relationship between feature vectors and arrival/service rates. More precisely, for every price $p \in [p_L, p_U]$, the firm can observe a feature vector $\mathbf{x}_p \in \mathbb{R}^{d_f}$. The arrival rate λ_p and the service rate μ_p have the following relationship with the feature vector $\mathbf{x}_p$:

$$\frac{1}{\lambda_p} = \theta_\lambda^T \mathbf{x}_p \text{ and } \frac{1}{\mu_p} = \theta_\mu^T \mathbf{x}_p, \tag{4.27}$$

where $\theta_\lambda \in \mathbb{R}^{d_f}$ and $\theta_\mu \in \mathbb{R}^{d_f}$ are two unknown coefficient vectors. Recall that $1/\lambda_p$ is the average time interval between two consecutive arrived customers and $1/\mu_p$ is the average service time under price p. That is, (4.27) assumes a linear relationship between the feature vector and the average arrival and service time. Similarly, we assume that the utilization factor $\rho_p = \frac{\lambda_p}{c\mu_p} < 1$ for any candidate price p, $\forall p \in [p_L, p_U]$ and Assumptions 4.2.1 and 4.2.2 hold.

Feature Vector in Practice. For practical applications, the feature vector can contain multiple factors that reflect the relationships between the prices and the service system parameters. For example, the feature factors can include demographic statistics related to customer segmentation, such as the number of nearby residents whose monthly income is above a certain times of the targeted price p. Another example is to use historical sales data to extract feature vectors. The historical data can be from another firm providing a similar service. This process for selecting relevant variables through data analysis is also termed by *feature engineering* (see an example for feature engineering in revenue management in Antonio et al. 2019).

Regret and Relaxed Regret. The regret defined in Definition 4.2.1 is valid for this continuous price setting. Under the above assumptions on this continuous price setting, we can readily check that Proposition 4.2.1 and Corollary 4.2.1 are still valid. Therefore, we can keep using the relaxed regret defined in Definition 4.2.2 as our performance measure of the proposed algorithm.

To solve this pricing problem with reusable resources under a continuous price setting, we propose an Online Batch LinUCB (BLinUCB) algorithm inspired by the general UCB algorithms for solving linear bandit problems with a large action space. There arises one major challenge of this paper: the linear bandit model of this revenue management problem with reusable resources has sub-exponential observations, which are caused by the Poisson customer arrival and service processes. Therefore, we develop concentration arguments for the sub-exponential observations under a linear relationship, and derive regret bound for this specific type of linear bandit problem with sub-exponential observations. To the best of our knowledge, our regret analysis provides the first theoretical study in linear bandits with sub-exponential observations (which can also be reviewed as sub-exponential rewards) and contributes to the broad landscape of linear bandits literature by delineating the impact of heavy-tailed random rewards.

4.5.1 Batch LinUCB Algorithm

Without loss of generality, we consider a known feature generation function $F(\cdot) : [p_L, p_U] \to \mathbb{R}^{d_f}$, which gives the firm the feature vector $\mathbf{x}_p$ of a price $p \in [p_L, p_U]$. We keep using notation $n_m(p)$ as the number of arrived customers under price p, $\hat{d}_i(p)$ as the observed time interval between the arrival time of customer i and that of customer $i-1$, $n_m^s(p)$ as the number of customers who have been successfully serviced, and $\hat{g}_i(p)$ as the observed service time of customer i.

4.5.1.1 M/M/c Queue Parameters under Linear Relationship

Consider a price $p \in [p_L, p_U]$ with a feature vector $\mathbf{x}_p$. We first analyze the arrival process. The arrival time interval is a random variable following an exponential distribution with mean $1/\lambda_p = \theta_\lambda^T \mathbf{x}_p$. Consider $n_m(p)$ observations of arrival times $\hat{d}_i(p),\ i = 1, \ldots, n_m(p)$ and denote $\bar{d}_p$ as the empirical mean of arrival time intervals $\bar{d}_p = \sum_{i=1}^{n_m(p)} \hat{d}_i(p)/n_m(p)$. Then the random variable $\bar{d}_p$ follows an Erlang distribution, more precisely, Erlang$(n_m(p), n_m(p)\lambda_p)$. For each implemented price p, we have a set of correspondent data with it, which can be represented by a tuple $(p, \mathbf{x}_p, \bar{d}_p, n_m(p))$.

Let SE denote sub-exponentials. By Lemma B.2.1, $\hat{d}_i(p) \sim$ SE$(4/\lambda_p^2, 2/\lambda_p)$. Then through Lemmas B.5.1 and B.5.2 (in Appendix B.5), we have

$$\bar{d}_p \sim \text{SE}(4/(n_m(p)\lambda_p^2), 2/(n_m(p)\lambda_p)).$$

Therefore, we can equivalently write it as

$$\bar{d}_p = \theta_\lambda^T \mathbf{x}_p + \epsilon_p, \tag{4.28}$$

where the random error term

$$\epsilon_p \sim \text{SE}\left(4/(n_m(p)\lambda_p^2), 2/(n_m(p)\lambda_p)\right).$$

Furthermore, by analyzing the Erlang distribution, we can derive that the mean of ϵ_p is 0 and the variance of ϵ_p is $1/(n_m(p)\lambda_p^2)$, where $1/\lambda_p = \theta_\lambda^T x_p$. Based on the weighted least squares approach (Seber and Lee 2012) for linear regression models with heteroscedasticity, we estimate the coefficient θ_λ as follows. Consider we have observations of N different prices and $N \geq d_f$. Define a diagonal matrix Ω where the ith element is the variance of the error term of the ith implemented price, i.e., $1/(n_m(p_i)\lambda_{p_i}^2)$. Note that λ_p is unknown and we use an approximate matrix $\hat{\Omega}$ to substitute Ω later. We use a matrix $\mathbf{X} \in \mathbb{R}^{N \times d_f}$ to denote the features of implemented prices and use a vector $\mathbf{d} \in \mathbb{R}^N$ to denote the empirical arrival time means of implemented prices. Then we can estimate the unknown coefficients by

$$\hat{\theta}_\lambda = (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{X}^T \Omega^{-1} \mathbf{d}. \tag{4.29}$$

With $N \geq d_f$, $\mathbf{X}^T \Omega^{-1} \mathbf{X}$ is non-singular and thus $\hat{\theta}_\lambda$ is well-defined. In addition, we can easily prove that $\mathbb{E}[\hat{\theta}_\lambda] = \theta_\lambda$ and Var$(\hat{\theta}_\lambda) = (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1}$. For any price p' with a feature vector $\mathbf{x}'$, we can estimate $\theta_\lambda^T \mathbf{x}'$ by $\hat{\theta}_\lambda^T \mathbf{x}'$ and further derive an upper confidence bound for $\theta_\lambda^T \mathbf{x}'$.

Proposition 4.5.1. ***(Sub-exponential Tail Bound for Poisson Process under Linear Relation-***

ship.) *Consider N implemented prices with $N \geq d_f$ and $n_m(p) \geq 8\log(T)$ for all implemented prices p. Then, for any new valid feature vector $\mathbf{x}'$ and $\hat{\theta}_\lambda$ computed in* (4.29)*, one has*

$$\mathbb{P}\left(\frac{|\hat{\theta}_\lambda^T\mathbf{x}' - \theta_\lambda^T\mathbf{x}'|}{\sqrt{\mathbf{x}'^T(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{x}'}} \geq \sqrt{32\log(T)}\right) \leq \frac{2}{T^4}. \tag{4.30}$$

Proof of Proposition 4.5.1. We first show the SE property of $\hat{\theta}_\lambda^T\mathbf{x}'$ and then derive the tail bound.

SE property of $\hat{\theta}_\lambda$. By the computing equation of $\hat{\theta}_\lambda$ in (4.29) and $\mathbf{d} = \mathbf{X}\theta_\lambda + \vec{\epsilon}$, we can derive

$$\hat{\theta}_\lambda = (\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}\mathbf{X}\theta_\lambda + (\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}\vec{\epsilon} = \theta_\lambda + (\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}\vec{\epsilon},$$

where $\vec{\epsilon} \in \mathbb{R}^N$ is a vector of all error terms ϵ_i of implemented prices in (4.28). Let matrix $A = (\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{X}^T\Omega^{-1}$ and thus $\hat{\theta}_\lambda = \theta_\lambda + A\vec{\epsilon}$. Use row vector $a_{d\cdot}$ to denote the dth row of A and a_{di} to denote elements of A. The dth element of $A\vec{\epsilon}$ is $a_{d\cdot}\vec{\epsilon}$, which follows $\text{SE}(\sum_{i=1}^N a_{di}^2(4/n_i\lambda_i^2), \max_{i=1,\ldots,N} 2|a_{di}|/(n_i\lambda_i))$ by Lemmas B.5.1 and B.5.2. Therefore, the dth element of $\hat{\theta}_\lambda$ follows $\text{SE}(\sum_{i=1}^N a_{di}^2(4/n_i\lambda_i^2), \max_{i=1,\ldots,N} 2|a_{di}|/(n_i\lambda_i))$ with mean $\theta_{\lambda,d}$.

SE property of $\hat{\theta}_\lambda^T\mathbf{x}'$. One has

$$\hat{\theta}_\lambda^T\mathbf{x}' = \mathbf{x}'^T(\theta_\lambda + A\vec{\epsilon}) = \mathbf{x}'^T\theta_\lambda + \mathbf{x}'^T A\vec{\epsilon}.$$

Then we focus on the second term. Similarly, use column vector $a_{\cdot i}$ to denote the ith column of A, and thus we have $\mathbf{x}'^T A = [\mathbf{x}'^T a_{\cdot 1}, \mathbf{x}'^T a_{\cdot 2}, \ldots, \mathbf{x}'^T a_{\cdot n}]$. By Lemmas B.5.1 and B.5.2, we can easily derive the second term follows $\text{SE}(\sum_{i=1}^N (\mathbf{x}'^T a_{\cdot i})^2(4/n_i\lambda_i^2), \max_{i=1,\ldots,N} 2|\mathbf{x}'^T a_{\cdot i}|/(n_i\lambda_i))$. By the definitions of matrices Ω and A, the first SE parameter can be rewritten as

$$\sum_{i=1}^N (\mathbf{x}'^T a_{\cdot i})^2(4/n_i\lambda_i^2) = 4\mathbf{x}'^T A\Omega A^T\mathbf{x}' = 4\mathbf{x}'^T(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{x}'.$$

Hence we can rewrite the SE distribution as $\text{SE}(4\mathbf{x}'^T(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{x}', \max_{i=1,\ldots,N} 2|\mathbf{x}'^T a_{\cdot i}|/(n_i\lambda_i))$. Therefore, $\hat{\theta}_\lambda^T\mathbf{x}'$ follows $\text{SE}(4\mathbf{x}'^T(\mathbf{X}^T\Omega^{-1}\mathbf{X})^{-1}\mathbf{x}', \max_{i=1,\ldots,N} 2|\mathbf{x}'^T a_{\cdot i}|/(n_i\lambda_i))$ with mean $\theta_\lambda^T\mathbf{x}'$.

Concentration bound for SE random variable $\hat{\theta}_\lambda^T\mathbf{x}'$**.** We apply the concentration inequality for SE variables. For SE variable X with parameters (ν^2, α), then $\mathbb{P}(|X - \mathbb{E}[X]| \geq t) \leq 2\exp(-t^2/(2\nu^2))$ if $t \leq \nu^2/\alpha$ (this concentration inequality can be found in, e.g., Boucheron et al.

2013). We can show

$$\left(\max_{i=1,\dots,N} \frac{|\mathbf{x}'^T a_{\cdot i}|}{n_i \lambda_i}\right)^2 = \max_{i=1,\dots,N} \frac{(\mathbf{x}'^T a_{\cdot i})^2}{n_i^2 \lambda_i^2} \leq \sum_{i=1,\dots,N} \frac{(\mathbf{x}'^T a_{\cdot i})^2}{n_i^2 \lambda_i^2} \tag{4.31a}$$

$$\leq \frac{1}{8\log(T)} \cdot \sum_{i=1,\dots,N} \frac{(\mathbf{x}'^T a_{\cdot i})^2}{n_i \lambda_i^2} = \frac{\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}{8\log(T)}, \tag{4.31b}$$

where the first inequality in (4.31b) is derived from $n_i \geq 8\log(T)$. Further, we have

$$\sqrt{32\log(T) \cdot \mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'} = \frac{4\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}{\frac{2\sqrt{\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}}{\sqrt{8\log(T)}}} \tag{4.32a}$$

$$\leq \frac{4\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}{\max_{i=1,\dots,N} 2|\mathbf{x}'^T a_{\cdot i}|/(n_i \lambda_i)}. \tag{4.32b}$$

Therefore, we meet the requirement for applying the sub-Gaussian behavior type of concentration bound and we have

$$\mathbb{P}\left(\frac{|\hat{\theta}_\lambda^T \mathbf{x}' - \theta_\lambda^T \mathbf{x}'|}{\sqrt{\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}} \geq \sqrt{32\log(T)}\right) \leq 2\exp(-\frac{32\log(T) \cdot \mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}{8\mathbf{x}'^T (\mathbf{X}^T \Omega^{-1} \mathbf{X})^{-1} \mathbf{x}'}) \leq \frac{2}{T^4}. \tag{4.33}$$

Q.E.D.

In the above analysis, we consider a known covariance matrix Ω. However, in the reality, this matrix is also unknown. Therefore, we estimate it by $\hat{\Omega}$ (also denoted by $\hat{\Omega}_\lambda$ for arrival process) according to the property of exponential distribution. We estimate the ith element, i.e., the variance of the error term of the ith implemented price, by $\bar{d}_{p_i}^2/(n_m(p_i)$. By the linear regression analysis with heteroscedasticity (see, e.g., Seber and Lee 2012), the property of estimated coefficient

$$\hat{\theta}_\lambda = (\mathbf{X}^T \hat{\Omega}_\lambda^{-1} \mathbf{X})^{-1} \mathbf{X}^T \hat{\Omega}_\lambda^{-1} \mathbf{d} \tag{4.34}$$

shares the same properties as the estimated coefficient in (4.29). Therefore, Proposition 4.5.1 also holds for $\hat{\theta}$ defined in (4.34). In the following analysis, we use $\hat{\Omega}$ to substitute Ω for both related parts to arrival and service processes.

We apply the same technique to the service process. Define $\hat{\Omega}_\mu$ and $\mathbf{y}_\mu$ similarly and we can compute an estimation as

$$\hat{\theta}_\mu = (\mathbf{X}^T \hat{\Omega}_\mu^{-1} \mathbf{X})^{-1} \mathbf{X}^T \hat{\Omega}_\mu^{-1} \mathbf{y}_\mu. \tag{4.35}$$

Therefore, we can reach the same result:

$$\mathbb{P}\left(\frac{|\hat{\theta}_\mu^T\mathbf{x}' - \theta_\mu^T\mathbf{x}'|}{\sqrt{\mathbf{x}'^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}'}} \geq \sqrt{32\log(T)}\right) \leq 2\exp\left(-\frac{32\log(T)\cdot\mathbf{x}'^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}'}{8\mathbf{x}'^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}'}\right) \leq \frac{2}{T^4}. \tag{4.36}$$

Proposition 4.5.2. *For price p with a feature vector $\mathbf{x}$, we have:*

$$\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)}}{\hat{\theta}_\lambda^T\mathbf{x}}\left(r_{\max}\sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}} + \sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\right) \geq 1 - \frac{4}{T^4}.$$

Proof of Proposition 4.5.2. The proof sketch is similar to that of Proposition 4.3.1.

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)}}{\hat{\theta}_\lambda^T\mathbf{x}}\left(r_{\max}\sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}} + \sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\right)\\
=&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}} + \frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)}}{\hat{\theta}_\lambda^T\mathbf{x}}\left(r_{\max}\sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}} + \sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\right)\\
\geq&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{r_{\max}\sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}}}{\hat{\theta}_\lambda^T\mathbf{x}}\right)\cdot\mathbb{P}\left(\left|\frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}}{\hat{\theta}_\lambda^T\mathbf{x}}\right).
\end{aligned}$$

We analyze the two terms separately. For the first term, we have

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{r_{\max}\sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}}}{\hat{\theta}_\lambda^T\mathbf{x}}\right)\\
=&\mathbb{P}\left(\frac{\lambda_p}{\mu_p\cdot\hat{\theta}_\lambda^T\mathbf{x}}\left|\theta_\lambda^T\mathbf{x} - \hat{\theta}_\lambda^T\mathbf{x}\right| \leq \frac{r_{\max}\sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}}}{\hat{\theta}_\lambda^T\mathbf{x}}\right)\\
\geq&\mathbb{P}\left(\left|\theta_\lambda^T\mathbf{x} - \hat{\theta}_\lambda^T\mathbf{x}\right| \leq \sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\\
\geq&1 - \frac{2}{T^4}.
\end{aligned}$$

For the second term, we has the following inequality by (4.36):

$$\mathbb{P}\left(\left|\frac{\theta_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}}{\hat{\theta}_\lambda^T\mathbf{x}}\right) = \mathbb{P}\left(\left|\theta_\mu^T\mathbf{x} - \hat{\theta}_\mu^T\mathbf{x}\right| \leq \sqrt{32\log(T)\,\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right) \geq 1 - \frac{2}{T^4}.$$

Therefore, we have

$$\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}}\right| \leq \frac{\sqrt{32\log(T)}}{\hat{\theta}_\lambda^T\mathbf{x}}\left(r_{\max}\sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}} + \sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\right) \geq (1 - \frac{2}{T^4})^2 \geq 1 - \frac{4}{T^4}.$$

Q.E.D.

4.5.1.2 Batch LinUCB

We present Batch LinUCB (BLinUCB) in Algorithm 4.3. Similarly, we divide the total horizon into two phases, *Warm-up* Phase and *Learning* Phase. In the Warm-up Phase, the algorithm gives a start of valid estimations of parameters θ_λ and θ_μ as computed in (4.34) and (4.35). To initiate invertible matrices $\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X}$ and $\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X}$, we select d_f number of prices, whose feature vectors form a basis for span$(\mathbf{x}_p, p \in [p_L, p_U])$, and collect $8\log(T)$ number of arrival and service time observations. Denote this set of basis prices as $\mathcal{P}_b$. In the Learning Phase, the algorithm separates the time as consecutive batches following the same definitions as those in BUCB and BTS, i.e., the length of batch $m = 1, \ldots, M$ is $I_m\tau$ with $I_m = 2^m$ and $\tau = \log(T)(\log\log(T))^2$. At the beginning of each batch, BLinUCB selects a price over the range $[p_L, p_U]$ with the highest upper confidence bound of the revenue rate, defined as:

$$U_c(p) = \left(\frac{\hat{\theta}_\mu^T\mathbf{x}}{\hat{\theta}_\lambda^T\mathbf{x}} + \frac{\sqrt{32\log(T)}}{\hat{\theta}_\lambda^T\mathbf{x}}\left(r_{\max}\sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\lambda^{-1}\mathbf{X})^{-1}\mathbf{x}} + \sqrt{\mathbf{x}^T(\mathbf{X}^T\hat{\Omega}_\mu^{-1}\mathbf{X})^{-1}\mathbf{x}}\right)\right)p. \quad (4.37)$$

Algorithm 4.3 Online Batch LinUCB Algorithm (BLinUCB).

```
Input: T, p_L, p_U, d_f.
Initialize: τ, I_m, M, P_b.
Warm-up Phase:
for p ∈ P_b do
    Offer price p, record d̂_i(p) for arriving customers and ĝ_i(p'), ∀p' ∈ [p_L, p_U] for leaving
    customers.
    if n^s_m(p) ≥ 8 log(T) then
        Update X, Ω̂_λ, Ω̂_μ, b_λ.b_μ
        Continue.
    end if
end for
Compute θ̂_λ and θ̂_μ by (4.34) and (4.35)
Learning Phase:
for m = 1, ..., M do
    Choose p_m = argmax_{p∈[p_L,p_U]} U_c(p).
    Offer p_m in batch m, i.e., for I_m τ periods.
    Record d̂_i(p_m) for arriving customers and ĝ_i(p), ∀p ∈ P for leaving customers.
    Update X, Ω̂_λ, Ω̂_μ, b_λ; Compute θ̂_λ and θ̂_μ.
end for
```

4.5.2 Performance Analysis of BLinUCB

We now quantify the performance of the BLinUCB algorithm. For notation simplicity, we use $\|\mathbf{x}\|_M = \sqrt{x^T M x}$ to denote the matrix norm of vector $\mathbf{x}$ induce by a positive semidefinite matrix M. Let $\mathcal{M}_{\lambda,m} = \mathbf{X}_m^T \hat{\Omega}_{\lambda,m}^{-1} \mathbf{X}_m$ and $\mathcal{M}_{\mu,m} = \mathbf{X}_m^T \hat{\Omega}_{\mu,m}^{-1} \mathbf{X}_m$, where the subscript m represents the results and information at the beginning of batch m. Therefore, at the beginning of batch m with a set of implemented prices $p = 1, \ldots, d_f + m - 1$, we can rewrite $U_c(p)$ as

$$U_c(p) = \left(\frac{\hat{\theta}_{\mu,m}^T \mathbf{x}}{\hat{\theta}_{\lambda,m}^T \mathbf{x}} + \frac{\sqrt{32\log(T)}}{\hat{\theta}_{\lambda,m}^T \mathbf{x}} \left(r_{\max} \|\mathbf{x}\|_{\mathcal{M}_{\lambda,m}^{-1}} + \|\mathbf{x}\|_{\mathcal{M}_{\mu,m}^{-1}} \right) \right) p \tag{4.38}$$

for any new price p with a feature vector $\mathbf{x}$. We also define the confidence radius of price p at the beginning of batch m as

$$\mathbf{Rad}_m^c(p) = \frac{\sqrt{32\log(T)}}{\hat{\theta}_{\lambda,m}^T \mathbf{x}} \left(r_{\max} \|\mathbf{x}\|_{\mathcal{M}_{\lambda,m}^{-1}} + \|\mathbf{x}\|_{\mathcal{M}_{\mu,m}^{-1}} \right) p.$$

For the sake of regret analysis, we shall further assume that the following assumptions hold.

Assumption 4.5.1. *The norm of covariates in $\{\mathbf{x}_p,\ p \in [p_L, p_U]\}$ is bounded: there exists $\mathcal{C}_\lambda, \mathcal{C}_\mu < \infty$ such that for all price $p \in [p_L, p_U]$, the corresponding feature vector $\mathbf{x}_p$ and the underlying arrival and service rates satisfy $\lambda_p \|\mathbf{x}_p\|_2 \le \mathcal{C}_\lambda$ and $\mu_p \|\mathbf{x}_p\|_2 \le \mathcal{C}_\mu$.*

Assumption 4.5.2. *The smallest eigenvalue of $\sum_{p \in \mathcal{P}_b} \lambda_m^2 \mathbf{x}_p \mathbf{x}_p^T$ is lower-bounded by l_λ and the smallest eigenvalue of $\sum_{p \in \mathcal{P}_b} \mu_m^2 \mathbf{x}_p \mathbf{x}_p^T$ is lower-bounded by l_μ.*

These two assumptions are mild, only requiring information about boundary cases of unknown system parameters. Assumption 4.5.1 can be satisfied if the firm knows upper bounds of the covariates of any context vectors $\|\mathbf{x}_p\|_2 \le \mathcal{C}_f$, any arrival rates $\lambda_p \le \lambda_{\max}$, and any service rates $\mu_p \le \mu_{\max}$. Then $\mathcal{C}_\lambda$ can take the value of $\lambda_{\max}\mathcal{C}_f$ and $\mathcal{C}_\mu$ can take the value of $\mu_{\max}\mathcal{C}_f$. Assumption 4.5.2 can be satisfied if the firm knows an lower bound l_f for the smallest eigenvalue of $\sum_{p \in \mathcal{P}_b} \mathbf{x}_p \mathbf{x}_p^T$, and lower bounds for any arrival rates $\lambda_p \ge \lambda_{\min}$ and any service rates $\mu_p \ge \mu_{\min}$. Then l_λ can take the value of $\lambda_{\min}^2 l_f$ and l_μ can take the value of $\mu_{\min}^2 l_f$. Both assumptions are used in the proof of Lemma B.6.2.

Theorem 4.5.1. *The T-period cumulative regret of BLinUCB is bounded by $O\left(d_f \log(T) \log\log(T) \sqrt{T}\right)$.*

For comparison, the state-of-the-art regret lower bound of linear bandits using UCB type of algorithms is of order $\Omega(d_f\sqrt{T})$ (see, e.g., Rusmevichientong and Tsitsiklis 2010). Therefore, this regret bound in Theorem 4.5.1 matches this lower bound up to a logarithmic factor.

Proof of Theorem 4.5.1. We bound the regret of Warm-up and Learning Phases separately. The notation definitions are the same as previous sections.

Regret of the Warm-up Phase. The difference between $J^*_{\text{Warm-up}}$ and $J^{\text{BLinUCB}}_{\text{Warm-up}}$ is at most linear on the length of the Warm-up Phase, which is $O(d_f \log(T))$. Therefore we can directly have

$$\overline{\text{Regret}}_{\text{Warm-up}}(\pi_{\text{BLinUCB}}, T) = O(d_f \log(T)). \tag{4.39}$$

Regret of nonstationarity during the Learning Phase. Follow the same analysis as those in the proof of Theorem 4.4.1, the expected revenue collected in batch m can be separated into two parts: the revenue collected during the transient state (i.e., the first 2τ periods) and during the steady state (i.e., the rest periods in this batch). Therefore, we can achieve a same result as (4.13). Therefore the relaxed regret of batch m can be bounded by

$$J^{\text{LP}}_{\text{Learning, m}} - J^{\pi_{\text{BLinUCB}}}_{\text{Learning, m}} \leq \Delta(p_m) I_m \tau + O(\tau).$$

Regret of suboptimality during the Learning Phase. We start by analyzing the regret of one batch. If the algorithm selects p_m for batch m, then we can still compute the time average regret of batch m (i.e., the loss of the revenue rate due to suboptimality) by (4.14). Furthermore, we can have $\Delta(p_m)$ bounded by

$$\min\Big\{\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_{c,m}(\tilde{p})\big) + \big(U_{c,m}(p_m) - L_{c,m}(p_m)\big) + \big(L_{c,m}(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\Big\} \tag{4.40a}$$

$$\leq \min\Big\{\big(U_{c,m}(p_m) - L_{c,m}(p_m)\big), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\Big\} + \big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_{c,m}(\tilde{p})\big) + \big(L_{c,m}(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\big), \tag{4.40b}$$

where (4.40a) is because the revenue rates are all non-negative. In the following proof we will consider the above three terms separately.

We further analyze the first term of (4.40b).

$$\begin{aligned}
&\min\Big\{\big(U_{c,m}(p_m) - L_{c,m}(p_m)\big), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\Big\} \\
=&\min\left\{2\mathbf{Rad}^c_m(p_m), \frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p}\right\} \\
=&\min\left\{2r_{\max}p_m\sqrt{32\log(T)}\left(\frac{1}{\hat{\theta}^T_{\lambda,m}\mathbf{x}}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\lambda,m}} + \frac{1}{r_{\max}\hat{\theta}^T_{\mu,m}\mathbf{x}_m}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\mu,m}}\right), \frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p}\right\} \\
\leq&\min\left\{2r_{\max}\tilde{p}\sqrt{32\log(T)}\left(\frac{1}{\hat{\theta}^T_{\lambda,m}\mathbf{x}_m}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\lambda,m}} + \frac{1}{r_{\max}\hat{\theta}^T_{\mu,m}\mathbf{x}_m}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\mu,m}}\right), r_{\max}\tilde{p}\right\} \\
\leq& r_{\max}\tilde{p}\cdot\min\left\{2\sqrt{32\log(T)}\left(\frac{1}{\hat{\theta}^T_{\lambda,m}\mathbf{x}_m}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\lambda,m}} + \frac{1}{r_{\max}\hat{\theta}^T_{\mu,m}\mathbf{x}_m}\|\mathbf{x}_m\|_{\mathcal{M}^{-1}_{\mu,m}}\right), 1\right\},
\end{aligned}$$

with $r_{\max} = \max_p \ \lambda_p/\mu_p$ and $\mathbf{Rad}_m^c(p_m)$ defined in (4.38). Take the expectation of both sides, we can derive that the expected value of this time average suboptimality loss is upper bounded by

$$\mathbb{E}\left[\min\left\{2\mathbf{Rad}_m^c(p_m), \frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}}\tilde{p}\right\}\right] \tag{4.41a}$$

$$\leq \mathbb{E}\left[r_{\max}\tilde{p}\cdot\min\left\{2\sqrt{32\log(T)}\left(\mathbb{E}\left[\frac{1}{\hat{\theta}_{\lambda,m}^T\mathbf{x}_m}\right]\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}} + \mathbb{E}\left[\frac{1}{r_{\max}\hat{\theta}_{\lambda,m}^T\mathbf{x}_m}\right]\|\mathbf{x}_m\|_{\mathcal{M}_{\mu,m}^{-1}}\right), 1\right\}\right] \tag{4.41b}$$

$$\leq \mathbb{E}\left[r_{\max}\tilde{p}\cdot\min\left\{2\sqrt{32\log(T)}\left(\lambda_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}} + \frac{\lambda_{p_m}}{r_{\max}}\|\mathbf{x}_m\|_{\mathcal{M}_{\mu,m}^{-1}}\right), 1\right\}\right] \tag{4.41c}$$

$$\leq \mathbb{E}\left[r_{\max}\tilde{p}\cdot\min\left\{2\sqrt{32\log(T)}\left(\lambda_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}} + \mu_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\mu,m}^{-1}}\right), 1\right\}\right]. \tag{4.41d}$$

Now we can analyze the cumulative regret by the end of the learning phase.

$$\overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BLinUCB}}, T) \tag{4.42a}$$

$$= \sum_{m=1}^{M}\left(J_{\text{Learning, m}}^{\text{LP}} - J_{\text{Learning, m}}^{\pi_{\text{BLinUCB}}}\right) \tag{4.42b}$$

$$\leq \sum_{m=1}^{M} I_m\tau\min\left\{\left(U_{c,m}(p_m) - L_{c,m}(p_m)\right), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\right\} + \sum_{m=1}^{M} I_m\tau\left(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - U_{c,m}(\tilde{p})\right) \tag{4.42c}$$

$$+ \sum_{m=1}^{M} I_m\tau\left(L_{c,m}(p_m) - r(\lambda_{p_m}, \mu_{p_m}, p_m)\right) + \sum_{m=1}^{M} O(\tau)$$

We first analyze the first term of (4.42c). Plug in the result in (4.41) we have:

$$\sum_{m=1}^{M} I_m\tau\min\left\{\left(U_{c,m}(p_m) - L_{c,m}(p_m)\right), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\right\}$$

$$\leq \sum_{m=1}^{M} I_m\tau\cdot r_{\max}\tilde{p}\cdot\min\left\{2\sqrt{32\log(T)}\left(\lambda_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}} + \mu_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\mu,m}^{-1}}\right), 1\right\} + O(\tau\log(T))$$

$$= r_{\max}\tilde{p}\cdot\sum_{m=1}^{M} I_m\tau\cdot\left(\min\left\{2\sqrt{32\log(T)}\lambda_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}}, 1\right\} + \min\left\{2\sqrt{32\log(T)}\mu_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\mu,m}^{-1}}, 1\right\}\right) + O(\tau\log(T)). \tag{4.43}$$

By Cauchy-Schwarz inequality, we can derive that

$$\sum_{m=1}^{M} I_m\tau \cdot \min\left\{2\sqrt{32\log(T)}\lambda_{p_m}\|\mathbf{x}_m\|_{\mathcal{M}_{\lambda,m}^{-1}}, 1\right\} \tag{4.44a}$$

$$\leq \sqrt{T}\cdot\sqrt{\sum_{m=1}^{M} I_m\tau \cdot \min\left\{128\log(T)\lambda_{p_m}^2\|\mathbf{x}_m\|^2_{\mathcal{M}_{\lambda,m}^{-1}}, 1\right\}} \tag{4.44b}$$

$$\leq \sqrt{T}\cdot\sqrt{\sum_{m=1}^{M} I_m\tau \cdot 128\log(T)\cdot \min\left\{\lambda_{p_m}^2\|\mathbf{x}_m\|^2_{\mathcal{M}_{\lambda,m}^{-1}}, \frac{1}{128\log(T)}\right\}} \tag{4.44c}$$

$$\leq \sqrt{T}\cdot\sqrt{\sum_{m=1}^{M} I_m\tau \cdot 128\log(T)\left(1+\frac{1}{128\log(T)}\right)\cdot \log\left(1+\lambda_{p_m}^2\|\mathbf{x}_m\|^2_{\mathcal{M}_{\lambda,m}^{-1}}\right)} \tag{4.44d}$$

$$= \sqrt{(128\log(T)+1)T}\cdot\sqrt{\sum_{m=1}^{M} I_m\tau \cdot \log\left(1+\lambda_{p_m}^2\|\mathbf{x}_m\|^2_{\mathcal{M}_{\lambda,m}^{-1}}\right)} \tag{4.44e}$$

$$= \sqrt{(128\log(T)+1)T}\cdot O\left(\sqrt{\sum_{m=1}^{M} n_m \cdot \log\left(1+\lambda_{p_m}^2\|\mathbf{x}_m\|^2_{\mathcal{M}_{\lambda,m}^{-1}}\right)}\right) \tag{4.44f}$$

$$\leq \sqrt{(128\log(T)+1)T}\cdot O\left(\sqrt{O(\tau)+2\mathcal{K}_\lambda d_f \log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8l_\lambda\log(T)}\right)}\right), \tag{4.44g}$$

where (4.44d) is by the fact that $x \leq (\alpha+1)\log(1+x)$ with $0 \leq x \leq \alpha$, (4.44f) is by $O(I_m\tau) = O(n_m)$, and (4.44g) is from Lemma B.6.1 (see Appendix B.6).

With (4.44), we can bound the regret in (4.43) as:

$$\begin{aligned}
&\sum_{m=1}^{M} I_m\tau \min\left\{\left(U_{c,m}(p_m) - L_{c,m}(p_m)\right), r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p})\right\} \\
\leq & r_{\max}\tilde{p}\cdot\left[\sqrt{(128\log(T)+1)T}\cdot O\left(\sqrt{O(\tau)+2d_f\mathcal{K}_\lambda\log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8l_\lambda\log(T)}\right)}+\right.\right. \\
& \left.\left.\sqrt{O(\tau)+2d_f\mathcal{K}_\mu\log\left(\frac{\mathcal{C}_\mu^2 O(T)}{8l_\mu\log(T)}\right)}\right)\right] + O(\tau\log(T)) \\
\leq & O\left(\log(T)\sqrt{T((\log\log(T))^2+d_f)}\right).
\end{aligned}$$

The analysis of the second and the third terms of (4.42c) are the same as (4.18) in the proof of Theorem 4.4.1, where here the results are derived from Proposition 4.5.2. We omit the details here.

Summing over the four terms of (4.42c) we bound the relaxed regret during the learning phase

as:

$$\begin{aligned}\overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BLinUCB}}, T) &= \sum_{m=1}^{M} \mathbb{E}\Big[\Delta(p_m) I_m \tau\Big] + \sum_{m=1}^{M} O(\tau) \\ &= O\left(\log(T)\sqrt{T((\log\log(T))^2 + d_f)}\right) + O\left(\frac{1}{T^3}\right) + O\big((\log(T))^2(\log\log(T))^2\big) \\ &= O\left(\log(T)\sqrt{T((\log\log(T))^2 + d_f)}\right).\end{aligned}$$

Total Regret of BLinUCB algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of BLinUCB algorithm as:

$$\overline{\text{Regret}}(\pi_{\text{BLinUCB}}, T) = \overline{\text{Regret}}_{\text{Warm-up}}(\pi_{\text{BLinUCB}}, T) + \overline{\text{Regret}}_{\text{Learning}}(\pi_{\text{BLinUCB}}, T) \tag{4.45a}$$

$$= O(d_f \log(T)) + O\left(\log(T)\sqrt{T((\log\log(T))^2 + d_f)}\right) \tag{4.45b}$$

$$= O\left(d_f \log(T) \log\log(T)\sqrt{T}\right). \tag{4.45c}$$

Q.E.D.

4.6 Extension to Settings with Balking and Reneging

We extend our model and algorithms to systems where customers may balk or renege when there is not enough amount of resources. We consider a discrete price set in this section. The same assumption holds for customer arrivals and the service processes, i.e., Poisson processes with rates λ_p and μ_p, respectively. Under any price p selected from the candidate price set $\mathcal{P}$, the service system can be reduced to an M/M/c queue model with balking and reneging (after finishing serving the customers arrived under previously offered prices). We consider a threshold $k_p \geq c,\ \forall p \in \mathcal{P}$. Upon arrival a customer joins the queue with probability one if the number of customers in the system is less than k_p. If there are k_p customers or more ahead of him/her, then he/she joins the queue with probability η_p and balks with probability $1 - \eta_p$. When the number of customers in the system is larger than k_p, the customers waiting in the queue may get impatient and then leave before being serviced. We assume that after a certain length of time, if the number of customers in the system is still larger than k_p, then one of the customers in the queue will leave the system. By assumption, this time follows an exponential distribution with mean $\frac{1}{\nu_p}$, or equivalently, the queue loses customers with rate ν_p when there are more than k_p customers in the system.

We assume that the threshold length of the queue for balking $k_p,\ \forall p \in \mathcal{P}$, is known. However, the arrival rate λ_p, service rate μ_p, the balking probability η_p, and the reneging rate ν_p are unknown

to the firm. We keep the same definition for the utilization factor of the system under price p, $\rho_p = \frac{\lambda_p}{c\mu_p}$, and the same assumption $\rho_p < 1,\ \forall p \in \mathcal{P}$. Therefore, the queue with balking and reneging has a smaller utilization of the buffer and thus also has a stationary distribution. Similarly, we assume that although the actual mappings between prices and system dynamics are unknown, a valid upper bound of ν_p/μ_p for all $p \in \mathcal{P}$ is known.

Assumption 4.6.1. *A constant $y_{\max}$ is known such that $y_{\max} \geq \max_{p\in\mathcal{P}} \nu_p/\mu_p$.*

For the remainder of this section, all notations are for the extended M/M/c queue with balking and reneging. We describe the model and regret for this extended system in §4.6.1. Then, the extended BUCB and BTS, with their performance analysis are presented in §4.6.2 and §4.6.3, respectively.

4.6.1 Regret for Extended Systems with Balking and Reneging

Proposition 4.6.1. *In the extended system with balking and reneging, the LP solution defined in* (4.2) *also provides an upper bound on the optimal revenue, i.e., $J^* \leq J^{LP}$.*

Proof of Proposition 4.6.1. Similar to the proof of Proposition 4.2.1, we show that the decisions variables (π, a, J) associated with any admissible policy must satisfy the constraints in (4.2), and the expected revenue of this admissible policy is exactly the corresponding objective value of (4.2).

Consider an arbitrary policy $\hat{\pi}$. We obtain the values of decision variables as follows. We follow the same procedure as in the proof of Proposition 4.2.1 to assign values to variables $\hat{a}_{nt}$, $\hat{\pi}^p_{nt}$, and $\hat{J}_{nt}$, for $n = 0, \ldots, \infty$, $t = 1, \ldots, T$, and $p \in \mathcal{P}$. Therefore, constraints (4.2b) and (4.2c) are immediately satisfied. Then we show that constraints (4.2d) are also satisfied. Without loss of generality, consider any period t where the policy chooses $\hat{p}$. The right-hand side of (4.2d) is the product of expected number of arrived customers $\lambda_{\hat{p}}$ and expected revenue from each customer $\frac{\hat{p}}{\mu_{\hat{p}}}$. For the left-hand side, the expected number of arrived customers is less than $\lambda_{\hat{p}}$ if balking or reneging happens and the average collected revenue from each customer can be less than $\frac{\hat{p}}{\mu_{\hat{p}}}$ if any arrived customers cannot finish the service before T. Therefore, constraints (4.2d) are also satisfied. To conclude, for this extended system, we also have $J^{LP} \geq J^{\pi}$, for any admissible policy π, and therefore for the state-dependent optimal policy π^*. **Q.E.D.**

Remark 4.6.1. *By the analysis of constraints* (4.2d) *in the proof of Proposition 4.6.1, we know that this upper bound J^{LP} for the extended system is looser than that for the system without balking and reneging. Due to this reason, we cannot bound the same relaxed regret to the state-dependent optimal policy, i.e., $J^{LP} - J^{\pi}$, for this extended system. To this end, we define a relaxed regret to the* state-independent *optimal policy as our performance measure for this extended system.*

Denote $\ell_{np},\ n = 0, \ldots, \infty$ as the probability that the system has n customers in the steady state

when offering price $p \in \mathcal{P}$, and thus $\sum_{n=0}^{\infty} \ell_{np} = 1,\ \forall p \in \mathcal{P}$. Let $\ell_p^{br} = \sum_{n=k_p+1}^{\infty} \ell_{np}$, and thus ℓ_p^{br} is the probability that balking and reneging may happen in the steady state. Since the arrival and the departure of the impatient customers without being serviced are independent, the average customer arrival rate can be expressed as:

$$\lambda_p - \big(\nu_p + (1-\eta_p)\lambda_p\big)\ell_p^{br}. \tag{4.46}$$

Therefore, one can derive the stationary revenue rate for the extended system as:

$$r^E(\lambda_p, \mu_p, \nu_p, \eta_p, p) = \frac{p}{\mu_p}\Big(\lambda_p - \big(\nu_p + (1-\eta_p)\lambda_p\big)\ell_p^{br}\Big). \tag{4.47}$$

Similarly, we refer to the price $\tilde{p} = \text{argmax}_{p\in\mathcal{P}} r^E(\lambda_p, \mu_p, \nu_p, \eta_p, p)$ as the state-independent optimal price. We derive an upper bound J^{IND} for the cumulative revenue collected by the optimal state-independent price $\tilde{p}$:

$$J^{\text{IND}} = T r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}). \tag{4.48}$$

Remark 4.6.2. *For the system without balking and reneging, we have $J^{LP} = J^{IND}$. For the extended system with balking and reneging, we have $J^{LP} = J^{IND} + T\big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \tilde{p}) - r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p})\big)$.*

Definition 4.6.1. *The relaxed regret to the state-independent optimal policy for system with balking and reneging is defined as*

$$\overline{Regret}^{IND}(\pi, T) = J^{IND} - J^{\pi}. \tag{4.49}$$

Next, we demonstrate the performance of the extended algorithms for the system with balking and reneging by analyzing the relaxed regret to the state-independent optimal policy.

4.6.2 BUCB and BTS for Extended Systems with Balking and Reneging

For the extended BUCB and BTS algorithms, we also record the following values by the end of each batch, in addition to the previously defined observations in § 4.3. We denote $n_m^c(p)$ as the number of arrived customers when the system has more than $k_p + 1$ customers; denote $n_m^b(p)$ as the number of arrived customers who choose to balk and $n_m^r(p)$ as the number of customers who join the queue but leave before getting serviced when offering p. For those customers who renege, we denote the observed waiting time that a customer get impatient and leave the queue by $\hat{h}_i(p)$ and then compute $\breve{h}_m(p) = \frac{n_m^r(p)}{\sum_{i=1}^{n_m^r(p)} \hat{h}_i(p)}$ as an empirical estimate of reneging rate. Let $\hat{\ell}_p^{br}$ denote the portion of the time when the number of customers in the system is greater than or equal to $k_p + 1$ after the first 2τ periods in each batch. Therefore, $n_m(p) \propto n_m^c(p) \propto n_m^b(p) \propto n_m^r(p),\ \forall m = 1, \ldots, M,\ p \in \mathcal{P}$. Then, we have an concentration inequality as follows, which is

similar to Proposition 4.3.1.

Proposition 4.6.2. *For any price $p \in \mathcal{P}$ by the end of batch m, $\forall m = 1, \ldots, M$, if $\min\{n_m^s(p), n_m^c(p), n_m^r(p)\} \geq 8\log(T)$. Then, we have*

$$\mathbb{P}\left(\left|\left(\frac{1}{\mu_p}\left(\lambda_p - (\nu_p + (1-\eta_p)\lambda_p)\ell_p^{br}\right)\right) - \left(\bar{g}_m(p)\left(\breve{d}_m(p) - \left(\breve{h}_m(p) + \frac{n_m^b(p)}{n_m^c(p)}\breve{d}_m(p)\right)\hat{\ell}_p^{br}\right)\right)\right| \leq \Gamma\right) \geq 1 - \frac{8}{T^3},$$

where

$$\Gamma = \frac{4(y_{\max} + r_{\max})}{T^3} + \left(8r_{\max} + 4y_{\max} + 9\breve{h}_m(p)\bar{g}_m(p) + 4\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p), n_m^r(p)\}}},$$

$$r_{\max} \geq \max_{p\in\mathcal{P}} \lambda_p/\mu_p, \qquad \text{and} \qquad y_{\max} \geq \max_{p\in\mathcal{P}} \nu_p/\mu_p.$$

The proof is derived based on concentration inequalities for sub-exponential random variables and Hoeffding's inequality for bounded random variables, and is provided in Appendix B.7. Similarly, we define the corresponding upper and lower confidence bound of the stationary revenue rate of the reduced queue of this extended system associated with price p by the end of batch m as follows.

Definition 4.6.2. *The upper confidence bound of the stationary revenue rate associated with price p by the end of batch m is:*

$$U_m^E(p) = \left(\bar{g}_m(p)\left(\breve{d}_m(p) - \left(\breve{h}_m(p) + \frac{n_m^b(p)}{n_m^c(p)}\breve{d}_m(p)\right)\hat{\ell}_p^{br}\right) + \Gamma'\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p), n_m^r(p)\}}}\right)p, \tag{4.50}$$

and the lower confidence bound of the stationary revenue rate associated with price p by the end of batch m is:

$$L_m^E(p) = \left(\bar{g}_m(p)\left(\breve{d}_m(p) - \left(\breve{h}_m(p) + \frac{n_m^b(p)}{n_m^c(p)}\breve{d}_m(p)\right)\hat{\ell}_p^{br}\right) - \Gamma'\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p), n_m^r(p)\}}}\right)p, \tag{4.51}$$

where $\Gamma'\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p), n_m^r(p)\}}} \geq \Gamma$.

Similarly, let $\mathbf{Rad}_m^E(p) = \Gamma'\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p), n_m^r(p)\}}}$ represent the confidence radius by end of batch m. Then for some $\gamma > 0$, the two properties, which are similar to (4.9) and (4.10), also hold:

$$\mathbb{E}[(r^E(\lambda_p, \mu_p, \nu_p, \eta_p, p) - U_m^E(p))^+] \leq \frac{\gamma}{PT}, \tag{4.52}$$

$$\mathbb{E}[(L_m^E(p) - r^E(\lambda_p, \mu_p, \nu_p, \eta_p, p))^+] \leq \frac{\gamma}{PT}. \tag{4.53}$$

4.6.2.1 Extended BUCB.

The extended BUCB is similar to the BUCB in Algorithm 4.1. At the beginning of each batch m, extended BUCB chooses price p with the highest $U_m^E(p)$ and offers this price during batch m, until reaching period T. We present the algorithmic details in Algorithm 4.4.

Algorithm 4.4 Extended Online Batch Upper Confidence Bound Algorithm (Extended BUCB).

Input: T, $\mathcal{P}$, k_p.
Initialize: τ, I_m, M.
Warm-up Phase:
for $p \in \mathcal{P}$ **do**
  Offer p, record $\hat{d}_i(p)$ for arriving customers, $n_m^b(p')$ for balking customers, $\hat{h}_i(p')$ for reneging customers, $\hat{g}_i(p')$ for leaving customers, $\forall p' \in \mathcal{P}$, and $\hat{\ell}_p^{br}$ for the system when the total customers is more than k_p.
  if $\min\{n_m^s(p), n_m^c(p), n_m^r(p)\} \geq 8\log(T)$ **then**
    Continue.
  end if
end for
Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $\breve{h}_m(p)$, $n_m(p)$, $n_m^c(p)$, $n_m^s(p)$, $n_m^r(p)$, $\forall p \in \mathcal{P}$.
Learning Phase:
for $m = 1, \ldots, M$ **do**
  Choose $p_m = \text{argmax}_{p\in\mathcal{P}} U_m^E(p)$.
  Offer p_m in batch m, i.e., for $I_m\tau$ periods.
  Record $\hat{d}_i(p_m)$ for arriving customers, $n_m^b(p)$ for balking customers, $\hat{h}_i(p)$ for reneging customers, $\hat{g}_i(p)$ for leaving customers, $\forall p \in \mathcal{P}$, and $\hat{\ell}_{p_m}^{br}$ for the system when the total customers is more than k_p.
  Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $\breve{h}_m(p)$, $n_m(p)$, $n_m^c(p)$, $n_m^s(p)$, $n_m^r(p)$, $\forall p \in \mathcal{P}$.
  Update $U_m^E(p)$, $\forall p \in \mathcal{P}$.
end for

4.6.2.2 Extended BTS.

For extended BTS, we similarly assume that the firm also holds a Gamma prior **Gamma**$(\alpha^{\nu_p}, \beta^{\nu_p})$ on ν_p for each price p, and the posterior distribution is **Gamma**$(\alpha^{\nu_p}+1, \beta^{\nu_p}+\hat{h}_i(p))$ after observing one time interval $\hat{h}_i(p)$ for reneging happens. The balking events can be viewed as Bernoulli trials, and thus we set the prior distribution of η_p as **Beta**$(\alpha^{\eta_p}, \beta^{\eta_p})$ and the posterior distribution of η_p is **Beta**$(\alpha^{\nu_p} + n_m^c(p) - n_m^b(p), \beta^{\eta_p} + n_m^b(p))$ after observing $n_m^b(p)$ out of $n_m^c(p)$ customers balk. To implement BTS, at the beginning of batch m, we sample parameters $(\hat{\lambda}_p, \hat{\mu}_p, \hat{\nu}_p, \hat{\eta}_p)$ following the updated posterior distributions and then offer price p with the maximum sampled stationary revenue rate $r^E(\hat{\lambda}_p, \hat{\mu}_p, \hat{\nu}_p, \hat{\eta}_p, p)$. We present the algorithmic details in Algorithm 4.5.

Algorithm 4.5 Extended Online Batch Thompson Sampling Algorithm (Extended BTS).

Input: T, $\mathcal{P}$, k_p, $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \alpha^{\nu_p},\ \beta^{\nu_p}, \alpha^{\eta_p},\ \beta^{\eta_p},\ \forall p \in \mathcal{P}$.
Initialize: τ, I_m, M.
Warm-up Phase:
for $p \in \mathcal{P}$ **do**
    Offer p, record $\hat{d}_i(p)$ for arriving customers, $n_m^b(p')$ for balking customers, $\hat{h}_i(p')$ for reneging customers, $\hat{g}_i('p)$ for leaving customers, $\forall p' \in \mathcal{P}$.
    if $\min\{n_m^s(p), n_m^c(p), n_m^r(p)\} \geq 8\log(T)$ **then**
        Continue.
    end if
end for
Update $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \forall p \in \mathcal{P}$.
Learning Phase:
for $m = 1, \ldots, M$ **do**
    Sample $\hat{\lambda}_p \sim$ **Gamma**$(\alpha^{\lambda_p}, \beta^{\lambda_p})$, $\hat{\mu}_p \sim$ **Gamma**$(\alpha^{\mu_p}, \beta^{\mu_p})$, $\hat{\nu}_p \sim$ **Gamma**$(\alpha^{\nu_p}, \beta^{\nu_p})$, $\hat{\eta}_p \sim$ **Beta**$(\alpha^{\eta_p}, \beta^{\eta_p})$, for $p \in \mathcal{P}$.
    Compute $\hat{\mathbb{P}}_m^{\text{br}}$ by plugging in $(\hat{\lambda}_p, \hat{\mu}_p, \hat{\nu}_p, \hat{\eta}_p),\ \forall p \in \mathcal{P}$.
    Choose $p_m = \text{argmax}_{p \in \mathcal{P}} r^E(\hat{\lambda}_p, \hat{\mu}_p, \hat{\nu}_p, \hat{\eta}_p, p)$.
    Offer p_m in batch m, i.e., for $I_m \tau$ periods.
    Record $\hat{d}_i(p_m)$ for arriving customers, $n_m^b(p)$ for balking customers, $\hat{h}_i(p)$ for reneging customers, $\hat{g}_i(p)$ for leaving customers, $\forall p \in \mathcal{P}$.
    Compute and update $\breve{d}_m(p)$, $\bar{g}_m(p)$, $\breve{h}_m(p)$, $n_m(p)$, $n_m^c(p)$, $n_m^s(p)$, $n_m^r(p),\ \forall p \in \mathcal{P}$.
    Update $\alpha^{\lambda_p},\ \beta^{\lambda_p},\ \alpha^{\mu_p},\ \beta^{\mu_p},\ \alpha^{\nu_p},\ \beta^{\nu_p},\ \alpha^{\eta_p},\ \beta^{\eta_p},\ \forall p \in \mathcal{P}$.
end for

4.6.3 Performance Analysis for Systems with Balking and Reneging

We analyze the relaxed regret to the state-independent optimal policy. The analysis follows the same roadmap as that for BUCB and BTS.

4.6.3.1 Coupling Analysis for M/M/c Queue with Balking and Reneging

More generally, we provide coupling analysis for any system that can be reduced to an M/M/c queue with balking and reneging satisfying the following assumption.

Assumption 4.6.2. *An arriving customer joins the queue with probability η_n and balks with probability $1 - \eta_n$ when n customers in the system; the queueing system loses customers with rate ν_n when there are n customers in the system:*

1. *η_n is non-increasing in n;*
2. *ν_n is non-decreasing in n.*

Assumption 4.6.2 holds in our setting in this section and various balking and reneging settings used in several literature, such as Ammar et al. (2013), Al-Seedy et al. (2009), Ward and Glynn (2003, 2005) and Zhang et al. (2005). Specifically, we refer the interested readers to a survey paper by Wang et al. (2010) about queueing systems with balking and reneging and a comprehensive summary by Ward (2012) about systems with reneging.

Lemma 4.6.1. ***(First-order Stochastic Dominance between A_i and A_j in M/M/c Queue with Balking and Reneging)*** *If Assumption 4.6.2 holds, then for random variables A_i and A_j where $i < j$, A_i first-order dominates A_j, i.e., $\mathbb{P}(A_j \leq x) \geq \mathbb{P}(A_i \leq x)$.*

Proposition 4.6.3. ***(Coupling Probability of M/M/c Queue with Balking and Reneging Starting from Empty State)*** *Consider Assumption 4.6.2 holds, the probability that the system with balking and reneging reaches the steady state within τ starting from an empty state is bounded from below by $1 - \frac{2}{T^3}$, where $\tau = \log(T)(\log\log(T))^2$.*

Proposition 4.6.4. ***(Coupling Probability of M/M/c Queue with Balking and Reneging when Price Changes)*** *Under Assumptions 4.2.2 and 4.6.2, the probability that the system with balking and reneging reaches the steady state within 2τ after the price changes is bounded from below by $1 - \frac{4}{T^3}$, where $\tau = \log(T)(\log\log(T))^2$.*

The proof of Lemma 4.6.1 is similar to the proof of Lemma 4.4.1, which is derived with the help of a virtual queueing system. We also develop stochastic dominance between busy periods of systems with or without balking and reneging. The proofs of Propositions 4.6.3 and 4.6.4 are similar to those of Propositions 4.4.1 and 4.4.2. We present the correspondent details in Appendix B.8.

4.6.3.2 Performance of Extended BUCB

Theorem 4.6.1. *The T-period cumulative regret of the extended BUCB algorithm (for systems with balking and reneging) can be bounded by $O(\sqrt{PT\log(T)})$.*

The proof of Theorem 4.6.1 is similar to the proof of Theorem 4.4.1. To bound the relaxed regret of the Learning Phase, we first bound the regret of suboptimality by deriving the difference of revenue rates by Proposition 4.6.2 and then we bound the regret of nonstationarity through Proposition 4.6.4. We relegate the full detailed proof to the Appendix B.8.

4.6.3.3 Performance of Extended BTS

Definition 4.6.3. *We define the relaxed Bayesian Regret to the state-independent optimal policy by period T as*

$$\overline{BR}^{IND}(\pi, T) = \mathbb{E}_{(\lambda,\mu,\nu,\eta)\sim prior}\big[\mathbb{E}[J^{IND} - J^{\pi}|\mathcal{F}_T]\big].$$

Theorem 4.6.2. *The T-period cumulative Bayesian regret of the extended BTS algorithm (for systems with balking and reneging) can be bounded by $O(\sqrt{PT\log(T)})$.*

The proof of Theorem 4.6.2 is similar to that of Theorem 4.4.2 and is given in Appendix B.8. The bound is derived from properties (4.52) and (4.53).

4.7 Numerical Experiments

We present the numerical results of BUCB and BTS under discrete price set (with balking and reneging) in §4.7.1, and then the numerical results of BLinUCB under continuous price set in §4.7.2. We also discuss the numerical results comparing discrete and continuous review settings in §4.7.3.

4.7.1 Numerical Results of BUCB and BTS with Discrete Price Set

Experimental Setup. The total operation time horizon is 1000 periods and the capacity of the reusable resource is $c = 100$. We consider three balking and reneging cases: (i) no balking and reneging happens, which means that the system under any price can be reduced to a standard M/M/c queue; (ii) the system is a loss system, where a newly arrived customer balks immediately when there is no available resource; (iii) both balking and reneging may happen: a newly arrived customer balks with probability 0.5 when there is no available resource and the reneging rate of the queue is 1, which means that one of the customers in the queue will get impatient and leave the queue after an uncertain time following an exponential distribution with mean 1. We choose the price from five candidate prices, of which the corresponding service rates are equal.

We consider three scenarios of the arrival rates associated with candidate prices and thus the corresponding system dynamics. In Scenario 1, the state-independent optimal price has a relatively low utilization factor. In Scenario 2, the state-independent optimal price has a relatively moderate utilization factor. In Scenario 3, the state-independent optimal price has the highest utilization factor. To conclude, we have nine instances in total (i.e., three scenarios under three cases), and the corresponding details are presented in Table 4.1.

Table 4.1: Parameter Settings and Experimental Design (Discrete Price Set)

	Price	Arrival Rate λ	Service Rate μ	Utilization Factor	Revenue Rate under Case (i)	Revenue Rate under Case (ii)	Revenue Rate under Case (iii)
					Instance #1	**Instance #2**	**Instance #3**
	18	5.5	0.1	0.55	990.00	988.95	989.37
	16	6	0.1	0.60	960.00	957.10	959.26
Scenario 1	14	6.5	0.1	0.65	910.00	907.91	909.81
	12	8	0.1	0.80	960.00	958.12	959.78
	10	9.5	0.1	0.95	950.00	902.37	910.91
					Instance #4	**Instance #5**	**Instance #6**
	18	5.2	0.1	0.52	936.00	935.20	935.39
	16	6	0.1	0.60	960.00	957.10	959.26
Scenario 2	14	7.5	0.1	0.75	1050.00	1049.47	1049.68
	12	8.5	0.1	0.85	1020.00	1009.27	1011.65
	10	9.5	0.1	0.95	950.00	902.37	910.91
					Instance #7	**Instance #8**	**Instance #9**
	18	5	0.1	0.50	900.00	898.96	899.06
	16	5.5	0.1	0.55	880.00	878.55	879.65
Scenario 3	14	6	0.1	0.60	840.00	837.46	839.35
	12	7.5	0.1	0.75	900.00	898.97	899.58
	10	9.8	0.1	0.98	980.00	915.88	921.04

Benchmark. We adopt the blind network revenue management algorithm proposed in Besbes and Zeevi (2012) to our settings as the benchmark, denoted by BZ benchmark for short. Algorithm 4.6 presents the algorithmic details of BZ benchmark which is implemented in this section. To conclude, for each instance, we implement five pricing algorithms: (extended) BUCB, (extended) BTS, and three benchmark policies with different values of the fraction of time allocated to the exploration phase, i.e., θ in Algorithm 4.6. We set θ as $0.22,\ 0.1$, and 0.05 in three BZ benchmarks respectively (the same as the settings in Table 1 in the original paper, Besbes and Zeevi 2012).

We compare the results of the above five pricing policies with that of the state-independent optimal price (i.e., "OPT" in all the figures). We present two figures for the results of each instance: the first shows the offered price over periods of each algorithm, $p_t,\ t = 1, \ldots, T$; the second depicts the cumulative time-average relaxed regret, including relaxed regret to the state-dependent

Algorithm 4.6 BZ Benchmark (adopted from Besbes and Zeevi 2012).

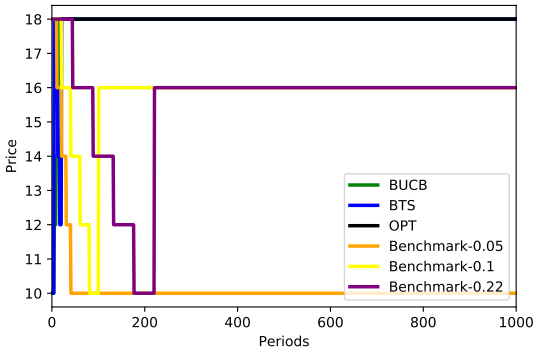
1: **Input:** T, $\mathcal{P}$, θ.
2: **for** $p \in \mathcal{P}$ **do**
3: Offer p for next $\lfloor(\theta \cdot T)/P\rfloor$ periods.
4: Collect revenue $\hat{J}_{p_m}$ from customers accepted with price p_m.
5: **end for**
6: Offer $p = \text{argmax}_{p\in\mathcal{P}} \hat{J}_p$ for the remaining periods until T.

optimal policy for Case (i) and relaxed regret to the state-independent optimal policy for Case (ii) and Case (iii), computed as the ratio between the relaxed regret and the time periods, i.e., $(\sum_{t'=1}^{t} J_{t'}^{\text{LP}} - \sum_{t'=1}^{t} J_{t'}^{\pi})/t$ for Case (i) and $(\sum_{t'=1}^{t} J_{t'}^{\text{IND}} - \sum_{t'=1}^{t} J_{t'}^{\pi})/t$ for Case (ii) and (iii). For Case (i), the relaxed regret to the state-dependent optimal policy is defined in Definition 4.2.2 and for Case (ii) and (iii), the relaxed regret to the state-independent optimal policy is defined in Definition 4.6.1. For each implemented policy, $\sum_{t'=1}^{t} J_{t'}^{\pi}$ is the revenue by the end of period t, which is collected from the simulation with the system starting with no customers at time 0. The results are presented in Figures 4.2–4.4.

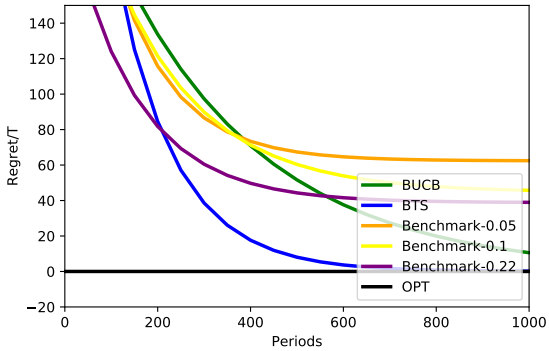
If the policy finally chooses the state-independent optimal price $\tilde{p}$, then the average regret over periods will shrink to 0 after a certain time, to offset the regret generated in the first several periods because of (i) choosing a sub-optimal price and (ii) the non-stationary transient states of the service system. If the policy finally chooses a sub-optimal price, then the average regret over periods will converge to a constant number, which theoretically is the difference between the stationary revenue rates of the chosen and the optimal state-independent price. In our theoretical proof, the regret analysis requires long time periods, i.e., a large T in Assumption 4.2.1. Our numerical results show that the proposed BUCB and BTS have good practical performance even when T is small; for example, we set $T = 1000$ in our experiments.

Compared with BZ. From the numerical results, we can see that both BUCB and BTS perform very well for all instances. They end up with the state-independent optimal price and the average regret converges to 0 quickly. The performance of three BZ benchmarks, parametrized with different θ, varies among different instances. In general, the benchmark with a larger θ has the higher probability of identifying the state-independent optimal price in the exploitation phase, but the performance is not guaranteed. One merit of our proposed policies is that the exploration and exploitation are all motivated by received information, and thus even when the policy chooses a sub-optimal price, it still has a chance to identify the correct one afterward. For example, in Instances #3 and #6, BUCB and BTS choose a sub-optimal price in the first several periods but identify and switch back to the state-independent optimal price later. However, the benchmarks cannot change the decision after entering the exploitation phase.

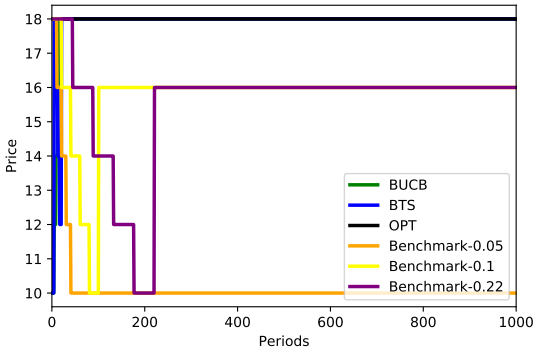
Converging Speed of Average Regret. The converging speeds of the three benchmarks highly



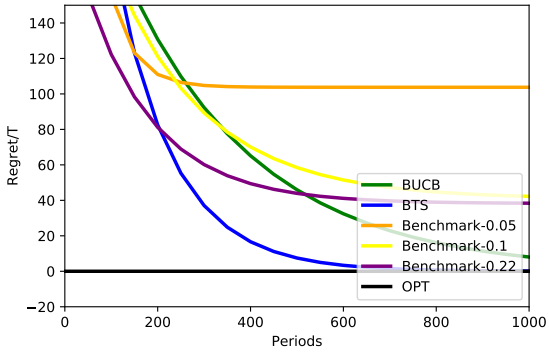
(a) The offered prices.



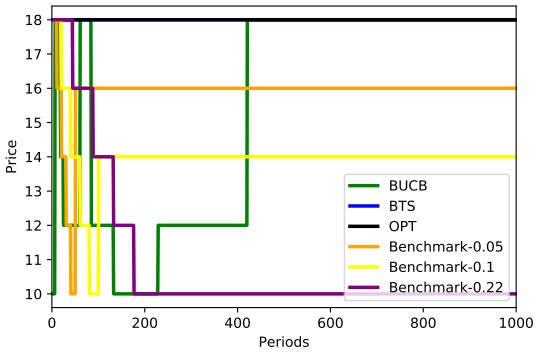
(b) The time-average cumulative regret.



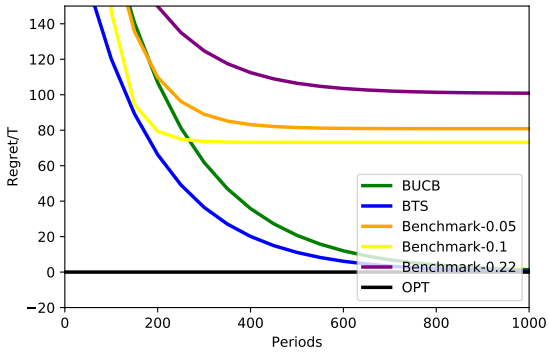
(c) The offered prices.



(d) The time-average cumulative regret.

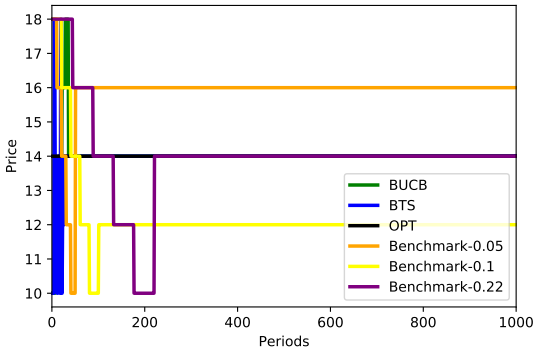


(e) The offered prices.

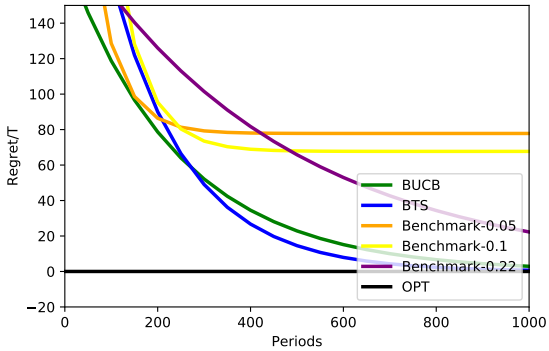


(f) The time-average cumulative regret.

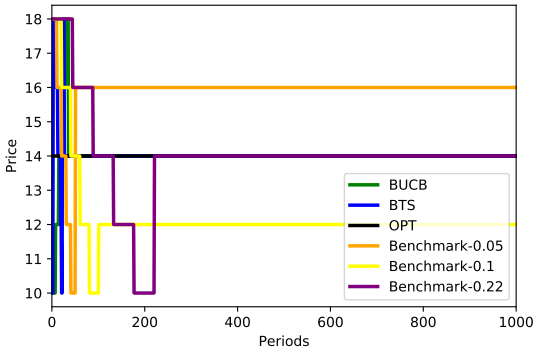
Figure 4.2: Offered Price and Time-average Cumulative Regret of Policies in Instances #1 to #3 (Rates-prices Scenario 1 Under Three Balking and Reneging Cases).



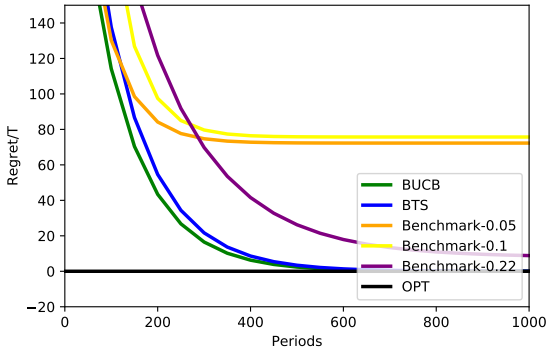
(a) The offered prices.



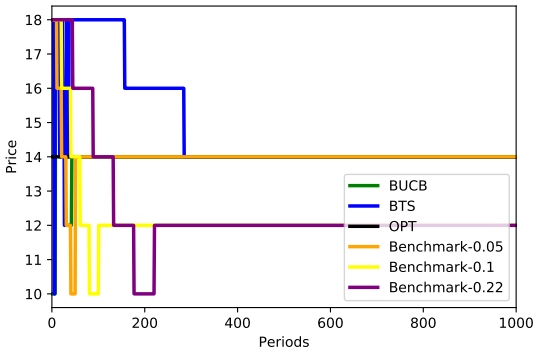
(b) The time-average cumulative regret.



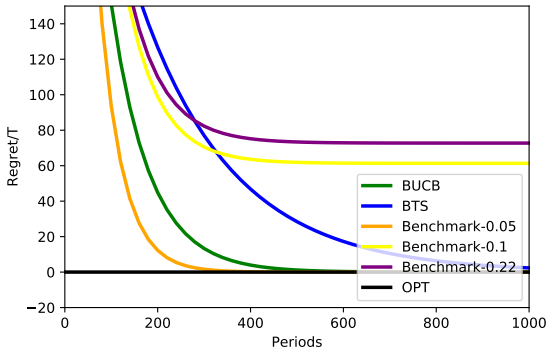
(c) The offered prices.



(d) The time-average cumulative regret.

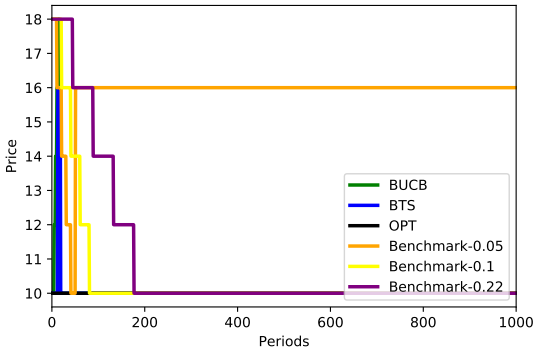


(e) The offered prices.

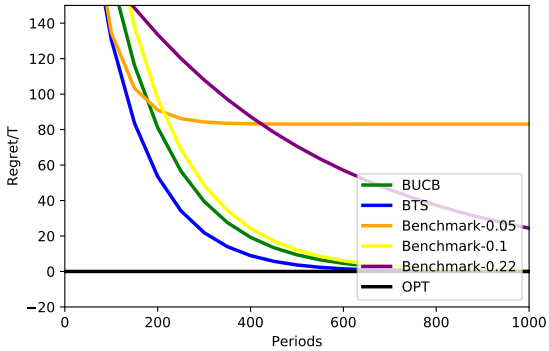


(f) The time-average cumulative regret.

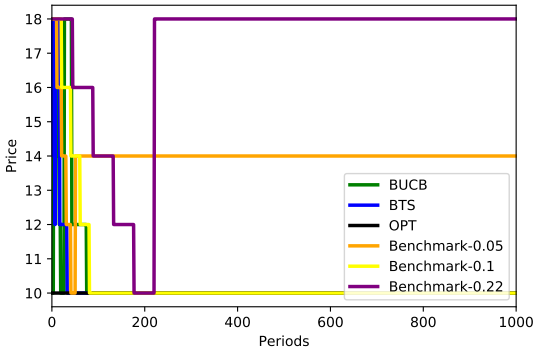
Figure 4.3: Offered Price and Time-average Cumulative Regret of Policies in Instances #4 to #6 (Rates-prices Scenario 2 Under Three Balking and Reneging Cases).



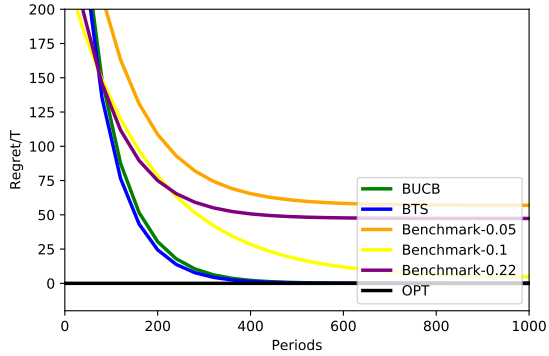
(a) The offered prices.



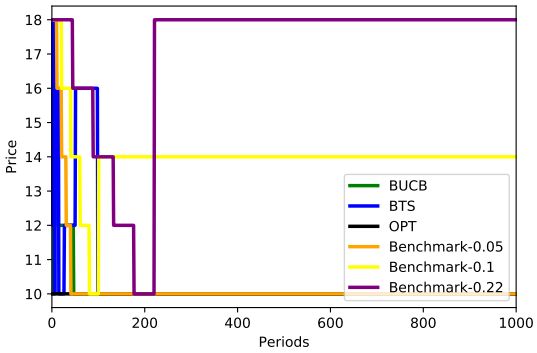
(b) The time-average cumulative regret.



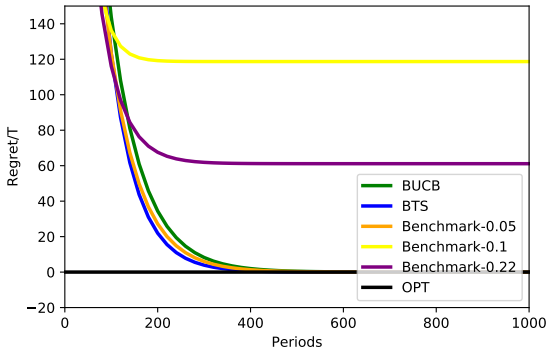
(c) The offered prices.



(d) The time-average cumulative regret.



(e) The offered prices.



(f) The time-average cumulative regret.

Figure 4.4: Offered Price and Time-average Cumulative Regret of Policies in Instances #7 to #9 (Rates-prices Scenario 3 Under Three Balking and Reneging Cases).

depend on the hyper-parameter, θ. A larger θ indicates that the benchmark spends a longer time in exploring the candidate prices and thus loses more potential revenue at the beginning. From the figures, we can see that the Benchmark-0.22 has the slowest converging speed among all three benchmarks. Comparing BUCB and BTS for all instances, we conclude that the converging speeds of BUCB and BTS are affected by the following two aspects. (1) The first aspect is the utilization factor of the state-independent optimal price. Comparing the performance of BUCB and BTS in Instances #1, #4, and #7, the converging speed of the regret over time for both the BTS and BUCB increases when the utilization factor of the optimal price increases. For Instance #7, the optimal price has the highest utilization factor, and the average regret of BUCB and BTS shirks to zero the fastest. The similar conclusion can be drawn by comparing Instances #2, #5, and #8 or comparing Instances #3, #6, and #9. (2) The second aspect is the existence of balking and reneging. Recall that Case (i) allows no balking and reneging while balking definitely happens in Case (ii). Therefore, we can consider that Case (i) and Case (ii) are two boundary cases of Case (iii). Comparing the converging performance of BUCB and BTS for Instances #1, #2, and #3, we observe that the converging speed of Case (i) is the slowest and the converging speed of Case (ii) is the fastest. One possible reason is that the system under Case (ii) reaches the steady-state faster than those under the other two cases and thus the two proposed pricing algorithms converge faster. The similar conclusion can be drawn by comparing Instances #4, #5, and #6 or comparing Instances #7, #8, and #9. And this property becomes more significant among Instances #7, #8, and #9. This is because if the utilization factor of the state-independent optimal price is higher, it is more likely the balking and reneging happen and consequently, the more significant difference among all three cases.

4.7.2 Numerical Results of BLinUCB with Continuous Price Set

Experimental Setup. The total operation time horizon is 8000 periods and the capacity of the reusable resource is $c = 100$. We choose the price from a fixed range $[10, 18]$, of which the corresponding service rates are equal. We consider a three-dimensional feature vector $(p, \phi(p), 1)$ for price p, where the second feature vector is defined as (we plot the value below in Figure 4.5):

$$\phi(p) = \frac{1}{1 + \exp\left(-\frac{5(2p-(p_L+p_U))}{p_U-p_L}\right)}.$$

Similar to the instance settings in §4.7.1, we consider three scenarios of the arrival rates associated with candidate prices and thus the corresponding system dynamics (three instances correspondingly). In Instance #10, the state-independent optimal price has a relatively low utilization factor.

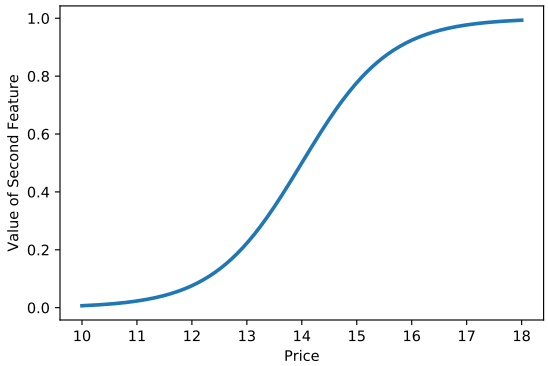
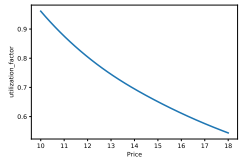
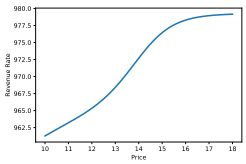
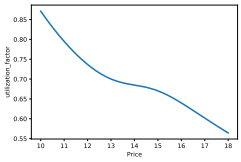
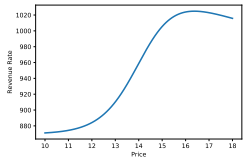
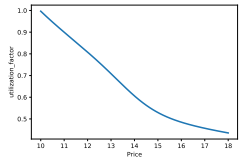
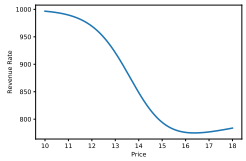


Figure 4.5: Function value of $\phi(p)$.

In Instance #11, the state-independent optimal price has a relatively moderate utilization factor. In Instance #12, the state-independent optimal price has the highest utilization factor. We present the instance details below in Table 4.2.

Table 4.2: Parameter Settings and Experimental Design (Continuous Price Range)

	Arrival Coef. θ_λ	Service Coef. θ_μ	Utilization Factor	Revenue Rate
Instance #10	$(0.0102, -0.0018, 0.0020)$	$(0, 0, 10)$		
Instance #11	$(0.0115, -0.03, 0)$	$(0, 0, 10)$		
Instance #12	$(0.01, 0.05, 0)$	$(0, 0, 10)$		

Benchmark. We adopt the ϵ-greedy algorithm (a commonly used benchmark that balances exploration and exploitation; see, e.g., Filippi et al. (2010)) to our settings as the benchmark. The ϵ-greedy algorithm will estimate the coefficients though a same regression step as that of BLinUCB. It will choose the price with the best estimated revenue rate with probability $1 - \epsilon$ and randomly select a price with probability ϵ. Further, we use the same batch framework as that of BLinUCB for

the benchmark and the algorithm makes pricing decision for each batch. Algorithm 4.7 presents the algorithmic details of ϵ-greedy benchmark we used in this section. To conclude, for each instance, we implement four pricing algorithms: BLinUCB and three benchmark policies with $\epsilon = 0.3$, 0.2, and 0.1, i.e., the probability for conducting exploration.

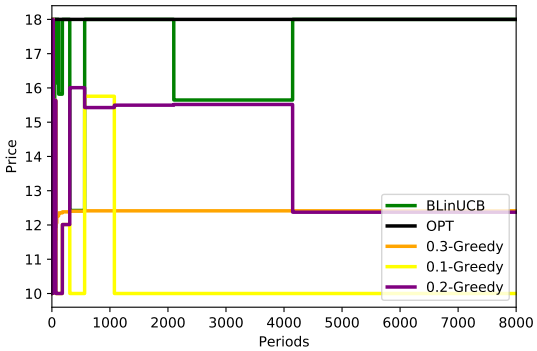
Algorithm 4.7 ϵ-greedy Benchmark.

Input: T, p_L, p_U, d_f, ϵ.
Initialize: τ, I_m, M, $\mathcal{P}_b$.
for $p \in \mathcal{P}_b$ **do**
  Offer price p, record $\hat{d}_i(p)$ for arriving customers and $\hat{g}_i(p'),\ \forall p' \in [p_L, p_U]$ for leaving customers.
  if $n_m^s(p) \geq 8\log(T)$ **then**
    Update $\mathbf{X}, \hat{\Omega}_\lambda, \hat{\Omega}_\mu, \mathbf{b}_\lambda.\mathbf{b}_\mu$
    Continue.
  end if
end for
Compute $\hat{\theta}_\lambda$ and $\hat{\theta}_\mu$ by (4.34) and (4.35)
for $m = 1, \ldots, M$ **do**
  With probability $1 - \epsilon$: Choose $p_m = \text{argmax}_{p \in [p_L, p_U]}\ p \cdot \frac{\hat{\theta}_\mu^T x_p}{\hat{\theta}_\lambda^T x_p}$.
  Otherwise, Choose $p_m \in [p_L, p_U]$ uniformly.
  Offer p_m in batch m, i.e., for $I_m \tau$ periods.
  Record $\hat{d}_i(p_m)$ for arriving customers and $\hat{g}_i(p),\ \forall p \in \mathcal{P}$ for leaving customers.
  Update $\mathbf{X}, \hat{\Omega}_\lambda, \hat{\Omega}_\mu, \mathbf{b}_\lambda$; Compute $\hat{\theta}_\lambda$ and $\hat{\theta}_\mu$.
end for

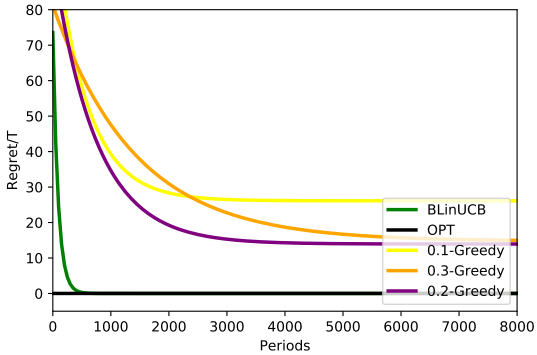
Similarly, we compare the results of the above four pricing policies with state-independent optimal price (i.e., "OPT" in all the figures). We present two figures for the results of each instance (see Figure 4.6): the first shows the offered price over periods of each algorithm and the second depicts the cumulative time-average relaxed regret, i.e., $(\sum_{t'=1}^{t} J_{t'}^{\text{LP}} - \sum_{t'=1}^{t} J_{t'}^{\pi})/t$.

Compared with ϵ-greedy Benchmarks. From the numerical results, we can see that BLinUCB performs the best for all instances. BLinUCB ends up with the state-independent optimal price in Instances #10 and #12, and with a near-optimal price in Instance #11. The difference of the revenue rate between the state-independent optimal price and the last price chosen by BLinUCB of Instance #11 is small, where the former is 1023 and the latter is 1020. One interesting observation is that all three ϵ-greedy benchmarks fail to identify the optimal price for these three instances. One possible reason could be the lack of exploration in the region where feature vectors are more vertical to those of implemented prices, resulting in inaccurate estimated coefficients of the linear relationship. The performance of three ϵ-greedy benchmarks, parametrized by ϵ, varies among different instances.

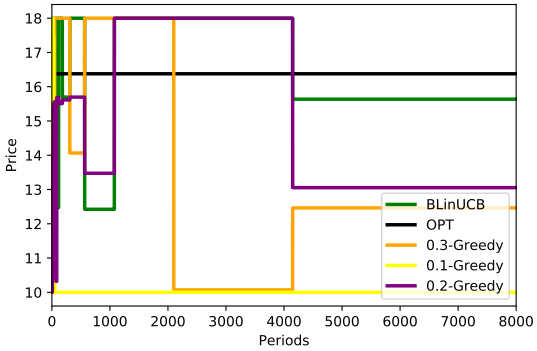
Converging Speed of Average Regret. For these three instances, we observe that the converging speed of the time average regret increases when the utilization factor of the optimal price



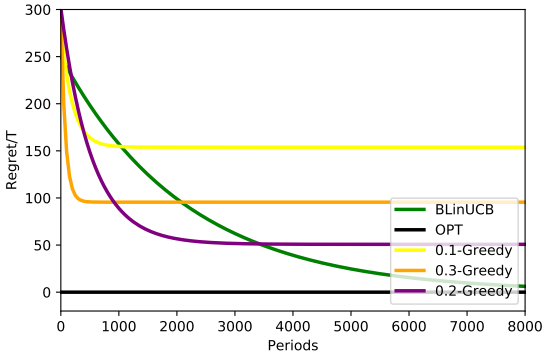
(a) The offered prices.



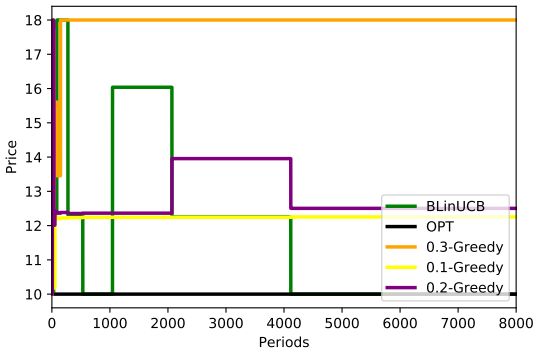
(b) The time-average cumulative regret.



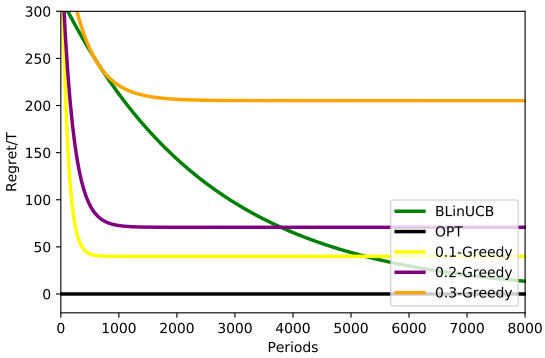
(c) The offered prices.



(d) The time-average cumulative regret.



(e) The offered prices.



(f) The time-average cumulative regret.

Figure 4.6: Offered Price and Time-average Cumulative Regret of Policies in Instances #10 to #12.

decreases. This seems counter-intuitive at first as the converging speed increases when the utilization factor of the optimal price increases in the discrete price setting. But a deeper thought reveals a potential reason: for the continuous price setting, a lower utilization factor of the optimal price implies higher utilization factors of sub-optimal prices, and the selection of these sub-optimal prices also helps achieve a more accurate estimation of the linear regression coefficients. This is not the case for the discrete price setting as the selection of other sub-optimal prices does not contribute to the estimation of the optimal price.

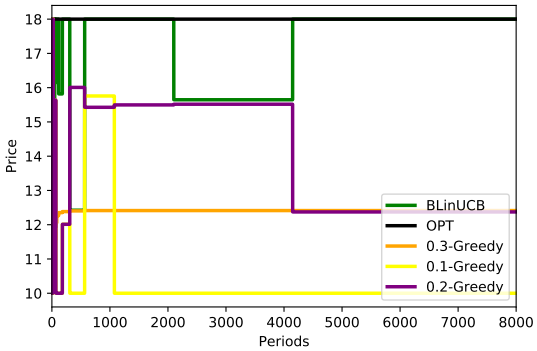
4.7.3 Continuous VS Discrete Time Control

We choose Instance #10 for testing the numerical performance of our algorithms facing different period lengths. The continuous time control setting for this revenue management problem can be seen as a "discrete" time setting where the length of the period is arbitrary close to 0. Therefore, we construct two additional variants of Instance #10 (termed #10.2 and #10.3). All three instances share the same horizon $T = 8000$. But the number of "discrete" periods is set to be 8000, 16000, and 80000, respectively. This corresponds to the length of a single period being 1, 0.5, and 0.1, respectively. We present the results of Instances #10, #10.2, and #10.3 in Figure 4.7.

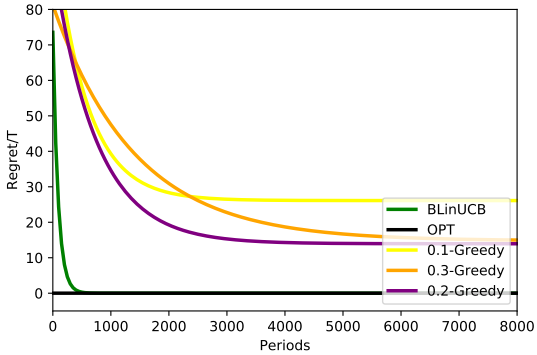
Recall that the baseline in the continuous time setting is still the LP value (see our Remark $4.2.2$). From the results of Instances #10, #10.2, and #10.3, we can observe that the time average regret (to the baseline LP value) of BLinUCB converges to zero quickly in all three instances. In addition, from the price selection results we can observe that the smaller period lengths do not significantly contribute to the identification of the optimal price. One possible reason is that although the system can subject to changes at any time, however, the actual changes of the states, i.e., the number of customers in the system, are discrete. This suggests that a discrete time setting with a reasonably small unit period length is good enough to obtain a near-optimal algorithm performance compared with the optimal policy under the continuous control setting. A practical guideline is to choose a unit period length smaller than the smallest among all mean interarrival time and service time.

4.8 Concluding Remarks

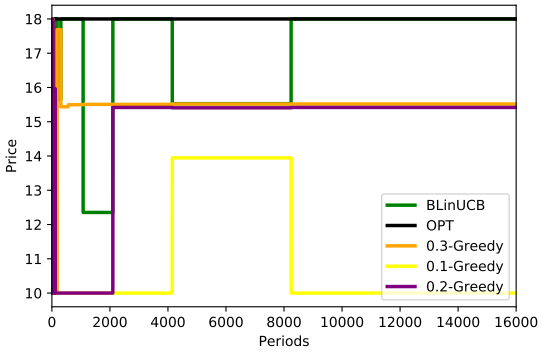
We consider a price-based revenue management problem with a single reusable resource over a finite time horizon. The firm serves stochastically arrived customers, who request an exponentially distributed service time and may balk and renege when there is not enough capacity of the resource. In this paper, we consider incomplete information, where the arrival rate, service rate, balking probability, and reneging rate all depend on the offered price, but the mappings are un-



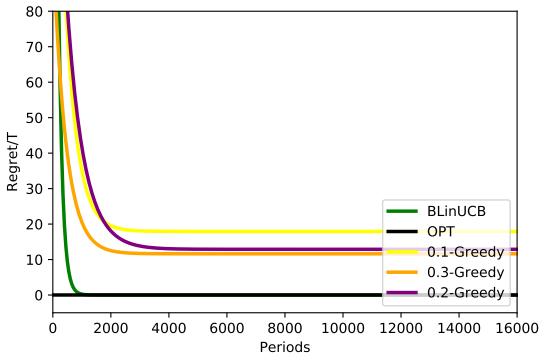
(a) The offered prices.



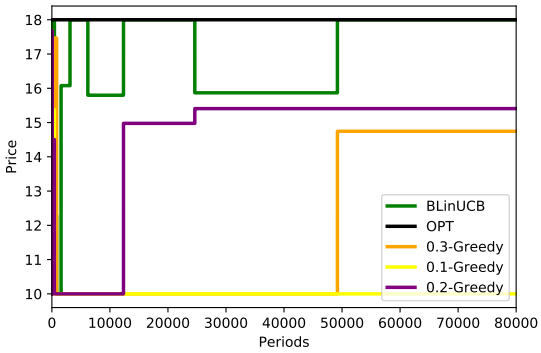
(b) The time-average cumulative regret.



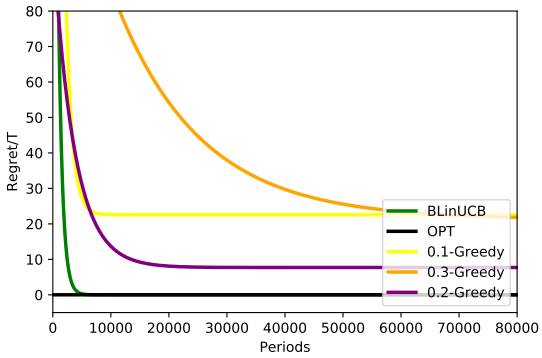
(c) The offered prices.



(d) The time-average cumulative regret.



(e) The offered prices.



(f) The time-average cumulative regret.

Figure 4.7: Offered Price and Time-average Cumulative Regret of Policies in Instances #10, #10.2, and #10.3.

known *a priori*. We develop two online learning and dynamic pricing policies to (i) learn the unknown mappings between the system dynamics and prices from past sales and (ii) maximize the cumulative revenue. We propose a Batch Upper Confidence Bound (BUCB) algorithm and a Batch Thompson Sampling (BTS) algorithm based on the multi-armed bandit (MAB) related algorithms. Our algorithms and regret analysis take into account following salient challenges, (i) the unknown rate-and-price mapping information, (ii) the dynamic nature of reusable resources being committed over time, (iii) the transient behavior of the service system when the price changes, and (iv) unbounded and heavy-tailed distributions of observable random variables. We prove that the cumulative regret is bounded from above by $O(\sqrt{PT\log(T)})$ under the discrete price setting and $O(d_f \log(T) \log\log(T)\sqrt{T})$ under the continuous price setting, where T is the total time periods, P is the number of the discrete prices, and d_f is the dimension of the feature vector. We also develop a new coupling analysis for analyzing the time for a system to reach the steady state from an empty state or from a steady state under another set of system parameters. Our numerical experiments demonstrate the performance of the proposed BUCB and BTS algorithms by showing that they can always identify the optimal state-independent price with a fast converging speed for the time-average cumulative regret.

There are several plausible future research avenues. First, this paper considers the single-resource setting for revenue management problems with reusable resources. Therefore, the demand learning and dynamic pricing under a multi-product and multi-resource setting is an interesting research problem (such as the setting considered in Owen and Simchi-Levi 2018, Doan et al. 2020). Second, one may consider generalizing the current model to accommodate general arrival and service distributions. However, this would require different methods in analyzing the regret, especially the loss of nonstationarity, which we shall leave it for future research. Third, this paper considers stationary demand and homogeneous personal activities in the system. There are also many literature considering heterogeneous and nonstationary demand (see, Borgs et al. 2014, Besbes et al. 2015, Lei and Jasin 2020). Therefore, extending the current model can benefit more real-world applications.

CHAPTER 5

Online Learning and Pricing for Network Service Systems with Reusable Resources

We consider a price-based network revenue management problem with multiple products and multiple reusable resources over a finite time horizon. Stochastically arrived customers request a product (service) that occupies a sequence of resources (servers). Each resource unit is occupied for an exponentially distributed service time and can be released to serve the next customer immediately, upon completion of serving one customer. The service-providing firm aims to construct a dynamic pricing control for all products to maximize the expected total revenue. In this paper, we assume that the firm does not know the mappings between customer demand across products and the offered prices, and thus it makes adaptive pricing decisions based only on past sales. Compared with prior network revenue management and multi-armed bandits literature, the salient difficulties of this problem lie in (i) the unknown demand-and-price mapping information, (ii) the unbalanced utilization ratios among resources, (iii) the dynamic nature of service systems with multiple servers being committed and released over time, (iv) the transient behavior of the service systems when the price changes, and (v) unbounded and heavy-tailed sub-exponential distributions of observed random variables. We propose two new multi-armed bandit (MAB) based learning algorithms for finding near-optimal pricing policies. Our proposed algorithms contain a Warm-up Phase to eliminate the heavy-tail effects, and a Learning Phase to identify the optimal price from a predefined candidate price set. Our algorithms further separate the Learning Phase into successive operational batches and adaptively select a price using past sales for the next batch. The performance measure is cumulative regret, which is the difference between the revenue attained by our approach and by a clairvoyant optimal pricing policy under full demand-and-price mapping information. We prove that both the proposed algorithms can achieve a cumulative regret $O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\log T\log(|\mathcal{J}|^{1/4}T)}\right)$, where $|\mathcal{J}|$ is the number of products, $|\mathcal{X}|$ is the cardinality of the feasible price set, and T is the total number of time periods. This regret result matches the lower bound up to a logarithmic factor. As an intermediate step of regret analysis, we are also the first to develop a coupling analysis for analyzing the time for a network queueing system to reach

the steady state from an empty state or from a steady state under another set of system parameters.

5.1 Introductory Remarks

Over the past two decades, network revenue management (NRM) problems (Gallego and Van Ryzin 1997) have been formulated and extensively studied. The proposed methods have achieved huge successes in a wide range of application areas including airline management, electric power management, fashion, healthcare system, and leisure (see, e.g., Williamson 1992, Talluri and Van Ryzin 2006, Özer et al. 2012). The bulk of existing literature focuses primarily on canonical settings with perishable resources in which the product once sold is permanently removed from inventory (see, e.g., Besbes and Zeevi 2012, Ferreira et al. 2018, Chen and Shi 2019b).

Apparently, network revenue management with reusable resources (RNRM) can also be used to model many real-life business and operations problems, including cloud computing services, hospitality industry, and leisure (Lei and Jasin 2020). In these applications, the firm is endowed with a finite number of different types of reusable resources to serve stochastically arrived customers over a finite selling horizon. Each customer requests a product, which uses a combination of resources for some random time, and release the occupied resources back to the firm, at which point the resources can be used to serve other customers.

For example, firms such as Amazon and Microsoft offer cloud computing services and users request to rent different types of computational resources, such as virtual machines, for a certain duration of time to finish their computational jobs (see, e.g., Kaewpuang et al. 2013, Püschel et al. 2015). The exact usage duration over each computational unit is not specified *a priori* and thus the departure of demand is also stochastic. The firm needs to dynamically decide what price to offer to users based on the current capacity utilization as well as the received requests (see, e.g., Doan et al. 2020). Another salient example arises from leisure business, such as amusement parks including Disneyland and Universal Studios. The firm is operating several theme parks and offers tickets for different combinations of the parks with various prices. Randomly arrived customers enter the parks and spend a random amount of time in each park. The firm needs to decide what price to charge for each type of tickets to maximize the total revenue, which is also unquestionably requiring a balance between the utilization of all parks. More examples can be found in health support industries, such as fitness center operations. In all the aforementioned applications, customers' willingness for using the diverse products is affected by the prices being offered; however, in reality, the decision-maker may not know the random demand distributions and how demand reacts to price changes, especially at the beginning of the planning horizon (see, e.g., Xu and Li 2013, Besbes and Zeevi 2012). Moreover, the total revenue not only depends on the customer demand but also the available resources for services, i.e., the changing real-time capacity

of servers.

This motivates us to study this complicated network revenue management with multiple reusable resources (servers) for providing multiple products (services). The firm is managing a finite number of different type of reusable resources with a fixed initial capacity and operating a finite number of products. The customers who requesting different products arrive according to a multivariate Poisson process (this arrival process is standard in the literature after the work by Gallego and Van Ryzin 1997). Each product represents a sequence of resources. A customer, who requesting a product, occupies one unit of the available resources following the correspondent order (or, passes through the correspondent sequence of servers). If there is no available next type of resource, when the customer finishes the usage of the previous resource in the sequence, she will join the queue of the next type of resource and wait. Each type of resources is occupied for an exponentially distributed consecutive time no matter which product it is serving for. Both the mean demand rate of each product and the average usage time for each resource depend on the posted price vector. Note that under a specific offered price vector, upon completion of existing customers under the previously posted price, the future service process of the system can be reduced to a queueing network consisting with M/M/c queues. The objective is to learn the mappings between customer demand rates and prices from past sales and offer the prices dynamically to maximize the expected total revenue collected over a finite time horizon.

5.1.1 Main Results and Contributions

We propose two new multi-armed bandit (MAB) based learning algorithms for network service systems with multiple reusable resources. The performance measure uses cumulative regret, which is the difference between the revenue attained by our learning algorithms and by a clairvoyant optimal pricing policy under full demand distributional information. We propose two algorithms, termed online batch Upper Confidence Bound for Network revenue management with Reusable resources (RNRM-UCB) and online batch Thompson Sampling for Network revenue management with Reusable resources (RNRM-TS). We prove in Theorems 5.5.1 and 5.5.2 that the cumulative regret is $O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\log T\log(|\mathcal{J}|^{1/4}T)}\right)$, where $|\mathcal{J}|$ is the number of products, $|\mathcal{X}|$ is the cardinality of the feasible price set, and T is the total number of time periods. This regret result matches the lower bound up to a logarithmic factor.

On a high level, our proposed algorithms contain a Warm-up Phase to eliminate the heavy-tail effects brought from the M/M/c queue observations and a Learning Phase to identify the optimal price from a predefined candidate price set. In the Learning Phase, we divide the planning horizon into multiple batches and iterate the prices over batches. We collect the observed demand information of all products and compute a revenue rate index for each candidate price. The price with

the highest index is chosen for next batch. One key difference (from the state-of-the-art literature in network revenue management with perishable resources, e.g., Ferreira et al. 2018, Chen and Shi 2019b) is that for each batch, we need to further divide it into two intervals. The first interval is for completing serving the existing customers (under previously posted price) and reaching the steady state of the corresponding Jackson network under the newly posted price. The system maintains the steady state in the second interval.

Compared with the prior network revenue management and MAB literature, there are several salient challenges when solving this problem.

(a) First, we consider the service system application scenario and thus one customer only occupies one unit of a specific reusable resource at the same time. Only when a customer finishes the usage on the previous resource (server), one unit of the next resource will be occupied and the previous resource unit is immediately released to serve next customer. In the other RNRM literature, the resources associated with a product is reserved for the customer until the customer leaving the system. Thus, the resources are keeping being committed and released over time and this results in a complicated queueing network. To tackle this difficulty, we thoroughly study the process for abstract the service system to a Jackson network and then we study the properties of Jackson network and develop correspondent theoretical results.

(b) Second, a major challenge for analyzing the collected revenue lies in bounding the ***loss of non-stationarity*** due to transient system performance. Indeed, as soon as the posted price changes, the arrival processes of each product type and the departure processes of newly arrived customers change. Moreover, the whole system, as well as each type of resources may serve customers who arrived during previous time periods under the old price and new customers who arrived in the current time period under the new price at the same time. Therefore, it is difficult to analyze the transient system performance and keep track of the cumulative revenue in a dynamic pricing scheme. To deal with this challenge, we are the first to develop a finite-time high probability bound on mixing times of Jackson network (see Propositions 5.5.1 and 5.5.2). As a result, we allow the system to take sufficient time to reach the steady state before changing the price.

(c) Third, the historical observations for updating the information, i.e., the interarrival time of customers of each product are exponentially distributed, thereby inducing ***unbounded and heavy-tailed random rewards***. This is in stark contrast with the sub-Gaussian assumption, which is generally required in the MAB literature. This light-tailed sub-Gaussian assumption is used to derive a tight concentration bounds for balancing the exploration and exploitation. We construct a Warm-up Phase to collection a certain amount of observations of each candidate

price. This construction invokes Bernstein's inequality and eliminates the negative effects of heavy-tailed sub-exponential random variables (see Proposition 5.4.1).

(d) Fourth, we need to compute an optimal pricing policy under the full information, which can only be derived in principle by exploiting the Markovian structure of the Jackson network and solving the associate Bellman equation. To achieve this goal, we derive a simple linear program (LP) for computing an optimal state-independent pricing decision. We show that the optimal objective value of this LP is an upper bound of the maximum cumulative revenue under full information scenario. Therefore, we can measure the performance of the proposed algorithms by comparing them with the optimal objective value of this LP, which is an upper bound of the general regret. In the analysis, we decompose the relaxed regret into two parts. The first part represents the revenue loss induced by the switching of prices, which is discussed in the aforementioned second challenge. The second part represents the ***loss of suboptimality***, which is induced by not offering the optimal state-independent price in certain time periods and can be bounded with the help of the solution to the aforementioned third challenge. This decomposition also stresses the trade-off between exploration and exploitation. Too much exploration on different prices would increase the regret of nonstationarity, and too little exploration would increase the regret of suboptimality. Our batched MAB approach finds the right and delicate balance.

5.1.2 Literature Review

Our work is closely related to following streams of literature.

5.1.2.1 Network Revenue Management with Reusable Resources.

Revenue Management with Demand Learning. Due to the accessibility of real-time sales data, there has been a large body of literature on developing learning and dynamic pricing algorithms with probably near-optimal regret upper bound for single-product single-resource setting (see, e.g., Besbes and Zeevi 2009, Araman and Caldentey 2009, Broder and Rusmevichientong 2012, Keskin and Zeevi 2014, Wang et al. 2014, Lei et al. 2014, den Boer 2015, Cheung et al. 2017, Chen and Chao 2019, Chen et al. 2020a) and NRM setting (see, e.g., Besbes and Zeevi 2012, Chen et al. 2019, Chen and Shi 2019b, Badanidiyuru et al. 2013, Agrawal and Devanur 2016, Ferreira et al. 2018). Specifically, Cheung et al. (2017), Chen and Chao (2019), Chen et al. (2020a) discussed practical reasons of infrequent price changes in revenue management and developed pricing algorithms with limited price switches. The algorithms developed in this paper also take the number of price switches into account by incorporating a batch design. All the aforementioned study consid-

ers the canonical settings (defined in Gallego and Van Ryzin 1994, 1997), where a firm endowed with non-reusable resources and the resources cannot be replenished throughout the season.

Revenue Management with Reusable Resources. There has recently been a growing body of literature focusing on revenue management with reusable resources. In order to dynamically match demands with the varying inventory of reusable resources, the first bulk of literature studies admission control and develops provably near-optimal heuristic controls with constant factor performance guarantees (see Levi and Radovanović 2010, Chen and Shi 2017, Chen et al. 2017). Besides admission control, a main stream of literature considering pricing problem, including this paper. Maglaras (2006) and Besbes et al. (2020) studied static pricing and the latter proved that a static pricing policy simultaneously guarantees 78.9% of the profit, market share, and service level from the optimal policy. Xu and Li (2013) and Doan et al. (2020) specifically studied revenue management problem with reusable resources in the application of cloud computing service providers and developed optimal and robust pricing policies to maximize the cumulative revenue. The former considers an exponential service time and the latter considers ambiguous distributions of the service time. The above literature are all consider single reusable resources and the only existing works on RNRM that consider multiple types of reusable resources are the recent works by Owen and Simchi-Levi (2018), Rusmevichientong et al. (2020) and Lei and Jasin (2020). Lei and Jasin (2020) also considered a multi-resource setting but under a deterministic service time assumption, and developed an asymptotically near-optimal real-time pricing control algorithm and further showed that their results can be extended to a more general setting with heterogeneous service time and advance reservation. Owen and Simchi-Levi (2018) considered a joint assortment and pricing problem under a multi-resource setting and provided constant factor performance guarantees for their proposed controls assuming discrete price points and exponential service time. Rusmevichientong et al. (2020) considered a dynamic assortment and pricing problem with reusable products and proposed a provably near-optimal policy based on approximate dynamic programming, which is guaranteed to obtain at least half of the optimal revenue.

Demand Learning with Reusable Resources. The aforementioned literature considering reusable resources all assumed that the distributional information of the underlying model is known to the decision-maker *a priori*. There has been very little work considering demand learning for service systems with reusable resources because of the complicated system dynamics. The only two consider demand learning for reusable resources are Chen et al. (2020b) and Chapter 4, and both of the work consider the single resource setting. Chen et al. (2020b) proposed an online learning framework for a dynamic pricing and capacity sizing problem in a GI/GI/1 queue and Chapter 4 developed MAB based algorithms for optimal dynamic pricing in an M/M/c queue.

Network Revenue Management with Reusable Resources and Demand Learning. Though network revenue management with reusable resources under incomplete demand information finds

a broad applications in today's economy, there has been no existing study in this area. To the best of our knowledge, we are the first to study network revenue management with reusable resources and demand learning.

5.1.2.2 Transient Analysis of Jackson Networks.

Jackson network has attracted a lot of attentions among the queueing networks because its steady state distribution is particularly simple to compute as a product-form solution (Jackson 1963, 1957). However, this simple product-form assumes steady state or it can be approximately achieved when the customers in the network is large enough (see, e.g., Mohamed et al. 2003). The transient analysis on single M/M/c queues have been conducted (Morisaku 1976, Kelton and Law 1985). Later, the queue lengths across the nodes in Jackson networks, i.e., the combination of single M/M/c queues, during the transient state can be bounded with the techniques of stochastic dominance (see Massey 1986, 1984, 1987). However, how the network reacts to system changes remains an open question (see, e.g., Mohamed et al. 2003, 2005). Kelton and Law (1985) pointed out that a queueing system need to run "long-enough" for getting rid off the effects brought by the initial starting point and then reach an ensured "steady-state" portion of the run. The only work that explicitly studies how quickly a queueing system can reach the steady state starting from an empty state of after system changes is Chapter 4. We proved that M/M/c queues can reach steady state within a certain amount of time, which is also referred to as the mixing time, with a high probability guarantee. To the best of our knowledge, the results in this paper give the first finite-time high probability bound on mixing times of Jackson networks (consisting of M/M/c queues) upon system parameter changes. We divide the nodes in a Jackson networks into layers and bound the mixing time layer by layer. Specifically, we establish a coupling argument between the target queueing system and a virtual system starting from a state sampled from the steady-state distribution and maintaining the steady state thereafter, when studying one specific node in the network.

5.1.2.3 Multi-Armed Bandit (MAB)

The MAB has been attracted extensive attentions and the literature predominantly assumes sub-Gaussian (including bounded) random rewards and proves tight regret bounds based on concentration inequalities exclusively for the light-tailed sub-Gaussian distributions (see, e.g., Ledoux 2001, Boucheron et al. 2013). The revenue management problem with reusable resources in this paper induces sub-exponential observations, one of heavy-tailed distributions, and thus the theoretical regret bound of the general MAB algorithms cannot be guaranteed by a direct adoption. Bubeck et al. (2013) was the first to thoroughly study MAB with heavy-tailed reward distributions. They considered rewards with moments of order $1 + \epsilon$ for some $\epsilon \in (0, 1]$ and replaced the empirical

mean in the upper confidence index with other robust estimators to obtain regret bounds of the same order as under sub-Gaussian reward distributions. Besides this work, there have been other studies devoted to MAB with heavy-tailed rewards using algorithmic structure and considering the cases under multi-play and multi-agent setting (see Vakili et al. 2013, Korda et al. 2013, Dubey and Pentland 2019, Xia et al. 2016, Dubey et al. 2020). A recent work by Jia et al. (2021) developed three UCB algorithms specifically to solve MAB with sub-exponential rewards. Our RNRM-UCB and RNRM-TS algorithms are employing a very similar framework as UCB-Warm in Jia et al. (2021) for allowing a Warm-up phase to cumulate certain amount of observations for each arm to achieve a well-defined upper confidence bound. Our problem is more challenging because each arm, i.e., each price in our problem, is associated with several types of observations, i.e., the product types.

As mentioned in Section 5.1.2.1, this price-based revenue management problem requires infrequent action changes. In the prolonged list of MAB literature, UCB2 (Auer et al. 2002a) and IMPROVED-UCB (Auer and Ortner 2010) were the first to develop M-batch policies with $M = \Theta(\log(T))$. Later, diverse studies on static batch design (see Perchet et al. 2016), adaptive batch design (see Gao et al. 2019), and explicitly model switching cost (see Cesa-Bianchi et al. 2013, Simchi-Levi and Xu 2019) have been conducted. In this paper, we consider a static batch design with $O(\log(T))$ number of batches, similar to UCB2 (Auer et al. 2002a) and the geometric grid design in Gao et al. (2019). We have this batched action framework to control the loss of the nonstationarity, whcih is resulted from the transient time after each action change.

In the revenue management area, some work also model revenue management problems as MAB problems and developed effective algirhtms accordingly. A recent work by Jia et al. (2020b) developed UCB and TS algorithms under single reusable resource setting and our paper considers multiple reusable resources and multiple products setting. For solving multiple resources setting, Badanidiyuru et al. (2013), Agrawal and Devanur (2016) proposed UCB based algorithms and (Ferreira et al. 2018) proposed TS based algorithms. However, these three work all consider non-reusable resources (NRM) while our paper considers reusable resources (RNRM).

5.1.3 Structure of the Paper

The remainder of this paper is organized as follows. In §5.2, we describe our RNRM problem under incomplete information. In §5.3, we define the regret and the relaxed regret. In §5.4, we describe the RNRM-UCB and RNRM-TS algorithms. In §5.5, we analyze the regret bound of RNRM-UCB and RNRM-TS algorithms, where we first provide in §5.5.1 a coupling analysis for Jackson network as an intermediate result of algorithmic performance analysis. In §5.6, we conclude the paper and point out several future research directions.

5.2 Problem Formulation

We consider a firm that manages $\mathcal{I}$ types of reusable resources and provides $\mathcal{J}$ types of services (we also refer to products interchangeably) to customers over a finite tine horizon T. Denote the capacity of server type $i \in \mathcal{I}$ as c_i. At the beginning of one period, the firm posts a price $\mathbf{x} = \{x_j : j \in \mathcal{J}\}$ for all services from a given price candidate set $\mathcal{X} \subset \mathbb{R}^{|\mathcal{J}|}$. Each arriving customer requests one service $j \in \mathcal{J}$, which is processed through an ordered sequence of servers. We further state the relationship between the sequence order of the consumed servers of each product in Assumption 5.2.1. Customers arrive following a Poisson process with rate $\alpha(\mathbf{x})$ and the service time of server i uniformally follows an exponential distribution no matter which product type it belongs, which is also depends on posted price $\mathbf{x}$. If there is not enough resource capacity, the customer will join the queue until being serviced. The servers are released to process next customer once finishing current customers. Customers pay x_j to the firm for product j. The goal of the firm is to maximize the total cumulative revenue.

Consider matrix $A_{ji} \in \mathbb{R}^{|\mathcal{J}|\times|\mathcal{I}|}$, where element $a_{ji} = 1$ indicates that service type j requires to be processed on server type i. We have the following assumption on the processing sequence of server types of each product.

Assumption 5.2.1. *There exists an order of server types $\mathcal{I}$ such that the processing sequence of servers $i_1, i_2, \ldots, i_n$ always has $i_1 < i_2 < \ldots < i_n$ for any product $j \in \mathcal{J}$.*

One can understand Assumption 5.2.1 from the view of a network system. If we consider a graph with server types as nodes and draw a directed arc from node i to i' if any product requires successive processing on server type i and i'. Then this graph is acyclic, i.e., no cycles, and there exists at least one topological order such that all arcs are from smaller index to larger index. Without loss of generality, we consider one topological order when referring to server i in the rest part of this paper. To better utilize this network representation of the service system, we also add a source node s to represent customers entering the system and a sink node t to represent customers existing the system. Therefore, the consuming matrix can be extended to $A_{ji} \in \mathbb{R}^{|\mathcal{J}|\times(|\mathcal{I}|+2)}$ and $a_{js} = a_{jt} = 1$ for all products. We refer readers to our illustrative example later for better a understanding of Assumptions 5.2.1 and 5.2.2 and system parameters.

Assumption 5.2.2. *Consider any server $i \in \mathcal{I}$ (which is always an intermediate node in the network representation). We divide the products that require to be processed on server i into subgroups by the server before occupy i (the previous server can be empty if i is the first server of this product). In each subgroup, we can further divide the products into subsubgroups by the next server i' after i (and similarly, the next server can be empty if i is the last server of this product). Then, we have the demand distribution of subsubgroups within each subgroup are the same.*

Assumption 5.2.2 assumes a memoryless property of the customer flows of different products

over all servers. If a new customer requests to use server i, then the probability that it requests to use server types $i+1, \ldots, |\mathcal{I}|$ does not depend on previously visited servers.

Assumption 5.2.3. *For each product $j \in \mathcal{J}$, there exists a "turnoff" price $\bar{p}_j < \infty$ such that, if the posted price is larger than this turnoff price, then the demand of this product is zero.*

Assumption 5.2.3 is also widely assumed in many multi-product and multi-resource network revenue management studies (see, e.g., Lei and Jasin 2020). This turnoff price can be used to effectively turn off demand of undesired products whenever needed. Therefore, for each candidate price $\mathbf{x} \in \mathcal{X}$, we can further decompose the products into turn-on product set $\mathcal{J}_{\text{on}}(\mathbf{x})$ and a turn-off product set $\mathcal{J}_{\text{off}}(\mathbf{x})$, where the latter can be empty.

We use $p_{ii'}$ to denote the product demand portion of subsubgroups corresponding to i' and thus $\sum_{i' \in \mathcal{I} \cup \{t\}} p_{ii'} = 1$. We call this matrix $\mathcal{P}$ the server routing matrix and it is also depending on the posted price. With Assumptions 5.2.1 and 5.2.2, the service system under any posted price $\mathbf{x} \in \mathcal{X}$ can be reduced to a Jackson Network consisting with $|I|$ number of M/M/c queues (when the system only contains customers arriving under this price, i.e., after finishing serving other customers arriving under the previously posted prices). We link this revenue management and Jackson Network as follows. The firm selects a price $\mathbf{x}$. (i) On the one hand, this price affects the future demand $\alpha(\mathbf{x})$ entering the system. (ii) On the other hand, this price affects the internal competition between the products, which is reflecting in the server routing matrix $\mathcal{P}(\mathbf{x})$. As a result, $\alpha(\mathbf{x})$ and $\mathcal{P}(\mathbf{x})$ define a Jackson Network well and the demand rate $\alpha_j(\mathbf{x})$ of each product type is also determined (use vector $\vec{\alpha}_{\mathbf{x}} \in \mathbb{R}^{|\mathcal{J}|}$ to denote demand of products). We use $\lambda_i(\mathbf{x})$ to denote the demand rate of server type i. We assume that the firm does not know the underlying mappings $\alpha(\mathbf{x})$ and $\mathcal{P}(\mathbf{x})$.

Assumption 5.2.4. ***Stability.*** *Though we do not explicitly consider the service time distribution, i.e., the service rate $\mu_i(\mathbf{x})$, of each server type $i \in \mathcal{I}$, we assume the queueing system of each service is stable under any price, i.e., $c_i\mu_i(\mathbf{x}) > \lambda_i(\mathbf{x})$. Therefore, the Jackson Network is stable as well.*

We use a $|\mathcal{J}|$-dimensional vector, $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$ to denote the number of customers across different types of servers. The firm aims at finding a pricing policy as $\pi : \{(\mathbf{y}, t) : \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ t = 1, \ldots, T\} \to \mathcal{X}$, where the firm selects price $\pi(\mathbf{y}, t) \in \mathcal{X}$ for period t when there are $\mathbf{y}$ customers in the system at the beginning of period t. The goal is to maximize the expected total revenue. The expected revenue under pricing policy π during period t, $\forall t = 1, \ldots, T$, is denoted by J_t^π and the cumulative expected revenue over periods $\{1, \ldots, T\}$ is denoted by $J^\pi = \sum_{t=1}^{T} J_t^\pi$.

In the following content, we refer to *servers* and *resources* interchangeably. We refer to *products* and *services* interchangeably.

We also consider the following assumptions for the arrival and service rate of each node (server)

under different candidate prices.

Assumption 5.2.5. ***Change of Rates.*** *For any node, consider the stable arrival and service rates* (λ_1, μ_1) *and* (λ_2, μ_2) *under any two candidate prices* $\mathbf{x}_1,\ \mathbf{x}_2 \in \mathcal{X}$ *where* $\lambda_1 \geq \lambda_2$*, also denote the utilization factor by* ρ_1 *and* ρ_2 *respectively, then:*

1. $\lambda_1 - \lambda_2 \leq \frac{\rho_1}{-3e \log \rho_1}$.
2. *if* $\mu_1 > \mu_2$*, then* $\rho_1 \geq \rho_2^2$.

Assumption 5.2.5 says the following: 1) the difference in arrival rates between two prices is bounded (by a large constant); and 2) if both the service and arrival rates under a price is higher than or equal to another price, the utilization factor in the former case must be greater than the latter squared. This condition (which is used to justify the requirement for applying an existing result in the proof of Proposition 5.5.2) is rather mild in the following sense: a simple sufficient condition to guarantee $\rho_l \geq \rho_h^2$ is $\rho_l \geq \rho_h$, which reduces to $\lambda_1/\lambda_2 \geq \mu_1/\mu_2$, meaning that the ratio of change in arrival rates (with respect to price) exceeds the ratio of change in service rates.

Assumption 5.2.6. *We assume* $\log\log(T) \geq 3$.

Assumption 5.2.6 says that we require the planning horizon to be sufficiently long, since it takes time for the underlying service system to reach a steady state under any posted price.

5.2.1 Illustrative Example

We refer to this example several times in the rest of the paper. Consider 4 types of servers and 4 types of products: Product No.1: $\{$Server 1 $\rightarrow$ Server 3$\}$, Product No.2: $\{$Server 1 $\rightarrow$ Server 3 $\rightarrow$ Server 4$\}$, Product No.3: $\{$Server 2 $\rightarrow$ Server 3$\}$, and Product No.4: $\{$Server 2 $\rightarrow$ Server 3 $\rightarrow$ Server 4$\}$. We can easily check that this network revenue management problem satisfies Assumption 5.2.1. We draw the network representation with the source and sink node in Figure 5.1. We assume this illustrative example satisfies Assumption 5.2.2. Therefore, if we consider the demand that flows out from the Server 3, we have that $\alpha_1/\alpha_2 = \alpha_3/\alpha_4 = p_{3t}/p_{34}$ holds under any price. We assume the demand function is $\alpha(\mathbf{x}) = 110 - \mathbf{x}^T\mathbf{1}$ and $p_{ii'} = (\sum_{j:a_{ji}=1,a_{ji'}=0} x_j)/(\sum_{j:a_{ji}=1} x_j)$. We consider a price $\mathbf{x} = (1, 2, 2, 5)$ and thus $\alpha(\mathbf{x}) = 100$. We present the system parameters

below.

$$\text{Consuming Matrix } A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \text{ Routing Matrix } P = \begin{cases} p_{s1} = \dfrac{x_3 + x_4}{x_1 + x_2 + x_3 + x_4} = 0.7 \\ p_{s2} = \dfrac{x_1 + x_2}{x_1 + x_2 + x_3 + x_4} = 0.3 \\ p_{3t} = \dfrac{x_2 + x_4}{x_1 + x_2 + x_3 + x_4} = 0.7 \\ p_{34} = \dfrac{x_1 + x_3}{x_1 + x_2 + x_3 + x_4} = 0.3 \\ p_{13} = p_{23} = p_{3t} = p_{4t} = 1 \end{cases},$$

$$\text{Product Demand: } \begin{cases} \alpha_1 = 0.7 \times 0.7 \times 100 = 49 \\ \alpha_2 = 0.7 \times 0.3 \times 100 = 21 \\ \alpha_3 = 0.3 \times 0.7 \times 100 = 21 \\ \alpha_4 = 0.3 \times 0.3 \times 100 = 9 \end{cases}, \text{ Server Demand: } \begin{cases} \lambda_1 = 0.7 \times 100 = 70 \\ \lambda_2 = 0.3 \times 100 = 30 \\ \lambda_3 = 1 \times 100 = 100 \\ \lambda_4 = 0.3 \times 100 = 30 \end{cases}.$$

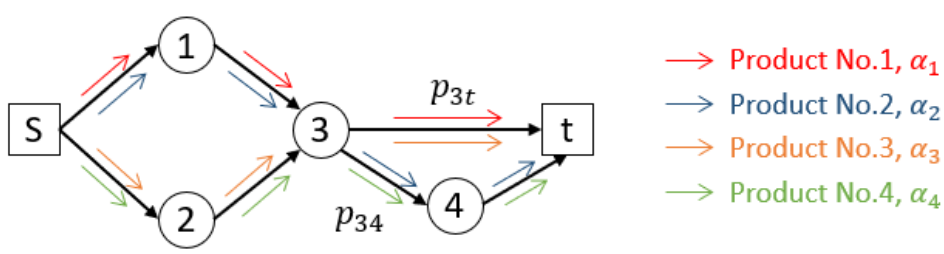


Figure 5.1: Servers and products in a network system.

5.3 Regret, Relaxed Regret and Static Price Benchmark

We use cumulative regret (see, e.g., Shalev-Shwartz et al. 2011) to evaluate the performance of proposed policies. We consider a full information scenario, where the firm knows the underlying arrival rate $\lambda_j(\mathbf{x})$ and service rate $\mu_j(\mathbf{x})$ of all products $j \in \mathcal{J}$ under any candidate price $\mathbf{x} \in \mathcal{X}$. We define a state-dependent optimal policy $\pi^* = \text{argmax}_\pi J^\pi$ and state-dependent optimal expected revenue $J^* = J^{\pi^*}$. Therefore, we can define the regret of any pricing policy π as the difference between the total state-dependent optimal expected revenue J^* and the total expected revenue under the given policy π, i.e., J^π. In short, we aim at finding heuristic pricing policies that lead to a small regret.

Definition 5.3.1. *The cumulative regret by the end of period T of policy π is defined as:*

$$\textit{Regret}(\pi, T) = J^* - J^\pi. \tag{5.1}$$

However, finding the state-dependent optimal pricing policy π^* is computationally intractable even under this full information scenario. Firstly, it requires to solve a dynamic program with an infinite number of potential states, in the form of $(\mathbf{y}, t)$, which suffers from the curse of dimensionality. Moreover, the transient dynamics of a queueing network is complex and hard to analyze especially when the offered price changes over periods. Thus, it is impractical or intractable to obtain the state-dependent optimal pricing policy π^*, and the corresponding optimal revenue J^*. To tackle this problem, we develop an upper bound on the expected revenue achievable by any policy, and therefore, also an upper bound of the state-dependent optimal expected revenue. The deriving process of the LP upper bound and Proposition 5.3.1 are similar to those in Jia et al. (2020b).

An LP upper bound. Consider continuous decision variables $\pi^{\mathbf{x}}_{\mathbf{y}t}$ for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$, $t = 1, \ldots, T$, $\mathbf{x} \in \mathcal{X}$, $a_{\mathbf{y}t}$ for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$, $t = 1, \ldots, T$, and $J_{\mathbf{y}t}$ for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$, $t = 1, \ldots, T$. We can formulate a linear program (LP) as follows.

$$J^{\text{LP}} = \max_{\pi, a, J} \sum_{\mathbf{y} \in \{0,1,\ldots,\infty\}^{|\mathcal{I}|}} \sum_{t=1}^{T} a_{\mathbf{y}t} J_{\mathbf{y}t} \tag{5.2a}$$

$$\text{s.t.} \sum_{\mathbf{y} \in \{0,1,\ldots,\infty\}^{|\mathcal{I}|}} a_{\mathbf{y}t} = 1 \quad \forall t = 1, \ldots, T \tag{5.2b}$$

$$\sum_{\mathbf{x} \in \mathcal{X}} \pi^{\mathbf{x}}_{\mathbf{y}t} = 1 \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T \tag{5.2c}$$

$$J_{\mathbf{y}t} \leq \sum_{\mathbf{x} \in \mathcal{X}} \pi^{\mathbf{x}}_{\mathbf{y}t} \cdot \vec{\alpha}^T_{\mathbf{x}} \mathbf{x} \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T \tag{5.2d}$$

$$0 \leq a_{\mathbf{y}t} \leq 1 \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T \tag{5.2e}$$

$$0 \leq \pi^{\mathbf{x}}_{\mathbf{y}t} \leq 1 \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T,\ \forall \mathbf{x} \in \mathcal{X}. \tag{5.2f}$$

Proposition 5.3.1. *The LP solution provides an upper bound on the optimal revenue, i.e., $J^* \leq J^{LP}$.*

Proof of Proposition 5.3.1. We show that the decisions variables (π, a, J) associated with any admissible policy must satisfy the constraints of the linear program (5.2), and the expected revenue of this admissible policy is exactly the corresponding objective value in (5.2). Consider an arbitrary policy $\hat{\pi}$. We obtain the values of decision variables by the following steps. Variable $\hat{a}_{\mathbf{y}t}$ takes the value of probability that the system has $\mathbf{y}$ customers at the beginning of period t under the policy $\hat{\pi}$ for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$ and $t = 1, \ldots, T$. Therefore, constraints (5.2b) are immediately satisfied. Variable $\hat{\pi}^{\mathbf{x}}_{\mathbf{y}t} = 1$ if the policy $\hat{\pi}$ chooses price $\mathbf{x}$ at state $(\mathbf{y}, t)$, otherwise, $\hat{\pi}^{\mathbf{x}}_{\mathbf{y}t} = 0$, for

$\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$, $t = 1, \ldots, T$, $\mathbf{y} \in \mathcal{X}$. As a result, constraints (5.2c) hold because in the left-hand side we have one variable equal to 1 and others all zero. Variable $\hat{J}_{\mathbf{y}t}$ takes the value of expected revenue collected before the end of period T from the customers who arrive during period t, when the system has $\mathbf{y}$ customers at the beginning of period t, for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$. Consider arbitrary $\mathbf{y}$ and t, we denote the price chosen at state $(\mathbf{y}, t)$ by the policy $\hat{\pi}$ as $\hat{\mathbf{x}}$. Then, by plugging in $\hat{\pi}^{\hat{\mathbf{x}}}_{\mathbf{y}t} = 1$, in the right-hand side of constraints (5.2d), we have $\vec{\alpha}^T_{\mathbf{x}}\mathbf{x}$, where $\vec{\alpha}^T_{\mathbf{x}}$ denotes the average number of arrived customers for consuming products during period t and $\hat{\mathbf{x}}$ is the revenue vector of these products. Therefore, $\vec{\alpha}^T_{\mathbf{x}}\mathbf{x}$ denotes the expected revenue collected from customers who arrive during period t (where assuming all arrived customers being served successfully). Consequently, we can conclude that the right-hand side of (5.2d) is as large as the left-hand side, because the right-hand side may also compute the revenue collected after the end of period T. Hence, constraints (5.2d) are also satisfied. By the value assignment steps of $\hat{a}_{\mathbf{y}t}$ and $\hat{J}_{\mathbf{y}t}$, for $\mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|}$, $t = 1, \ldots, T$, one has that the corresponding LP objective value at solution $(\hat{\pi}, \hat{a}, \hat{J})$ is $J^{\hat{\pi}}$. Therefore, we have $J^{LP} \geq J^{\pi}$, for any admissible policy π, and thus also for the optimal state-dependent policy π^*. **Q.E.D.**

By defining $\tilde{\mathbf{x}} = \text{argmax}_{\mathbf{x} \in \mathcal{X}} \vec{\alpha}^T_{\mathbf{x}}\mathbf{x}$, we find an optimal solution $(\tilde{\pi}, \tilde{a}, \tilde{J})$ to the LP in (5.2) as follows.

$$\tilde{\pi}^{\tilde{\mathbf{x}}}_t = 1, \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T \tag{5.3a}$$

$$\tilde{\pi}^{p}_{\mathbf{y}t} = 0, \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|},\ \forall t = 1, \ldots, T,\ \forall \mathbf{x} \in \mathcal{X} \backslash \{\tilde{\mathbf{x}}\} \tag{5.3b}$$

$$\tilde{J}_{\mathbf{y}_0 t} = \frac{\lambda_{\tilde{\mathbf{x}}}}{\mu_{\tilde{\mathbf{x}}}}\tilde{\mathbf{x}} \quad \forall t = 1, \ldots, T \tag{5.3c}$$

$$\tilde{J}_{\mathbf{y}t} = 0 \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|} \backslash \{\mathbf{y}_0\},\ \forall t = 1, \ldots, T \tag{5.3d}$$

$$\tilde{a}_{\mathbf{y}_0 t} = 1 \quad \forall t = 1, \ldots, T \tag{5.3e}$$

$$\tilde{a}_{\mathbf{y}t} = 0 \quad \forall \mathbf{y} \in \{0, 1, \ldots, \infty\}^{|\mathcal{I}|} \backslash \{\mathbf{y}_0\},\ \forall t = 1, \ldots, T, \tag{5.3f}$$

where $\mathbf{y}_0 = [0]^{|\mathcal{J}|}$. Then, we can compute

$$J^{\text{LP}} = T \cdot \vec{\alpha}^T_{\mathbf{x}}\mathbf{x}. \tag{5.4}$$

Note that $\vec{\alpha}^T_{\mathbf{x}}\mathbf{x}$ is the stationary revenue rate of price $\mathbf{x}$, denoted as $r(\mathbf{x})$. We call the price $\tilde{\mathbf{x}}$ as the (static) optimal state-independent price, i.e., the price with the highest stationary revenue rate. This result can be seen as the "reusable analog" of the classical static pricing upper bound for revenue management with perishable resources (see Gallego and Van Ryzin 1994).

Definition 5.3.2. *The relaxed regret is defined as*

$$\overline{Regret}(\pi, T) = J^{LP} - J^{\pi}. \tag{5.5}$$

Based on Proposition 5.3.1, we have for any policy π, the corresponding relaxed regret is a valid upper bound of its regret, i.e., $\overline{\text{Regret}}(\pi, T) \geq \text{Regret}(\pi, T)$. In the following sections, we demonstrate the performance of proposed algorithms by analyzing the relaxed regret.

5.4 Algorithms

In this section, we develop two algorithms to solve this network revenue management problem with reusable resources under incomplete demand mapping information scenario, termed online batch Upper Confidence Bound for Network revenue management with Reusable resources (RNRM-UCB) and online batch Thompson Sampling for Network revenue management with Reusable resources (RNRM-TS). We start with brief introductions of UCB and TS algorithms for solving general MAB problems.

UCB. The stochastic MAB problems consider i.i.d. random rewards for each arm and the player aims to find the arm resulting in the highest expected reward. The concept of confidence bound firstly appeared in Lai and Robbins (1985) and the first version of UCB algorithm was proposed in Lai (1987). The UCB algorithm defines an upper confidence bound of the expected reward of each arm based on the past observations and the actual expected reward is highly likely below this bound, supported by a probabilistic concentration inequality. Therefore, this upper confidence bound can be viewed as an optimistic estimate and thus guide an adaptive exploration. UCB1 in Auer et al. (2002b) and Slivkins et al. (2019) chooses the arm with the highest upper confidence bound and there are also many UCB variants (see, e.g., Garivier and Moulines 2011). The UCB algorithms achieve better performance compared with non-adaptive exploration based algorithms. We refer to Chapter 7 in Lattimore and Szepesvári (2020) and Chapter 3 in Slivkins et al. (2019) for more technical details and concrete summaries of UCB algorithms. Our RNRM-UCB is developed based on UCB1.

TS. This randomized Bayesian algorithm was firstly proposed by Thompson (1933, 1935) and it considers the case that the player also hold priori information of the uncertain rewards. The MAB problems under this Bayesian assumption is referred to as Bayesian MAB (see, e.g., Chapter 3 in Slivkins et al. 2019). While UCB type of algorithms keep track of the UCB of each arm, TS type of algorithms keep track of the posterior distribution associated with each arm. At each round, random samples are generated following the posterior distributions of each arm, and then the arm with the highest sampled reward is chosen. TS is also known as the "probability matching" property

because the probability of an arm being chosen by the algorithm matches the posterior probability that this arm has the highest expected reward. TS gradually becomes a representative algorithm for solving Bayesian MAB because it achieves similar theoretical performance guarantees while often better empirical performance (see Kaufmann et al. 2012, Chapelle and Li 2011), and it has been adapted to various Bayesian bandit problem settings (Russo and Van Roy 2014). We refer interested readers to Russo et al. (2018) and Agrawal and Goyal (2013) and two books by Slivkins et al. (2019) and Lattimore and Szepesvári (2020) for technical details of TS. A similar work to our paper is by Ferreira et al. (2018), where the authors considered NRM with perishable resources.

Both UCB and TS balance the exploration-exploitation trade-off and we proposed two dynamic pricing algorithms that builds on these simple, yet powerful, machine learning techniques. In Section 5.4.1, we derive an upper confidence bound of the steady-state revenue rate of each price after eliminating the heavy-tailed effects brought by the exponential observations of this revenue management. In Sections 5.4.2 and 5.4.3, we provide the algorithmic details for implementing the RNRM-UCB and RNRM-TS algorithms, respectively.

5.4.1 Probabilistic Concentration Bound

The basis for implementing a UCB type of algorithm is identifying an upper confidence bound of the expected collected revenue over a certain amount of periods under each candidate price. Thus there rises two challenges. The first challenge is the intractability of computing the expected revenue of this complicated network service system. We tackle this difficulty by firstly studying the steady-state revenue rate. Then with the results in Section 5.5.1 we can show that this steady-state revenue rate can be used to compute a lower bound for the expected revenue collected over an interval of consecutive periods. The second challenge is to derive an upper confidence bound for the steady-state revenue rate based on historical sales information. Jia et al. (2020b) established a concentration bound of the steady-state revenue rate under the single product setting. In this section, we develop a concentration bound for constructing an upper confidence bound of the steady-state revenue rate for the multi-product multi-resource setting, which also depends on the number of products.

We consider the heavy-tailed properties of exponential distributed inter-arrival time, induce by the Jackson network, into consideration and derive an upper confidence bound for the steady-state revenue rate in Proposition 5.4.1. We denote $n_j(\mathbf{x})$ as the number of arrived customers requesting product j under price $\mathbf{x}$. For each customer k arrived under the price $\mathbf{x}$, we denote $\hat{d}_{jk}(\mathbf{x})$ as the observed time interval between the arrival time of customer k and that of customer $k-1$ who are requesting product j. We use $\bar{d}_j(\mathbf{x})$ to denote the empirical mean of the inter-arrival time, $\bar{d}_j(\mathbf{x}) = \sum_{k=1}^{n_j(\mathbf{x})} \hat{d}_{jk}(\mathbf{x})/n_j(\mathbf{x})$, for $j \in \mathcal{J}$.

Proposition 5.4.1. *For any price $\mathbf{x} \in \mathcal{X}$, if the number of arrival time observations for each turn-on product $n_j(\mathbf{x}) \geq 8\log(|\mathcal{J}|^{\frac{1}{4}}T)$ for $j \in \mathcal{J}_{on}(\mathbf{x})$, then we have*

$$\mathbb{P}\left(\left|\vec{\alpha}_{\mathbf{x}}^T\mathbf{x} - \sum_{j\in\mathcal{J}_{on}(\mathbf{x})}\frac{x_j}{\bar{d}_j(\mathbf{x})}\right| \leq \sum_{j\in\mathcal{J}_{on}(\mathbf{x})}\frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right) \geq 1 - \frac{2}{T^4}.$$

Proof of Proposition 5.4.1. The proof is based on the concentration bound for sub-exponential random variables. For product $j \in \mathcal{J}_{\text{off}}$, their demand is off, i.e., $\alpha_j(\mathbf{x}) = 0,\ j \in \mathcal{J}_{\text{off}}(\mathbf{x})$. In this proof, we only consider one price and thus we use α_j to represent $\alpha_j(\mathbf{x})$ for notation simplicity. We can rewrite the right-hand-side as:

$$\begin{aligned}
&\mathbb{P}\left(\left|\vec{\alpha}_{\mathbf{x}}^T\mathbf{x} - \sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\frac{x_j}{\bar{d}_j(\mathbf{x})}\right| \leq \sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right)\\
=&\mathbb{P}\left(\left|\sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\left(\alpha_j x_j - \frac{x_j}{\bar{d}_j(\mathbf{x})}\right)\right| \leq \sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right)\\
\geq&\mathbb{P}\left(\sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\left|\alpha_j x_j - \frac{x_j}{\bar{d}_j(\mathbf{x})}\right| \leq \sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right)\\
\geq&\prod_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})}\mathbb{P}\left(\left|\alpha_j x_j - \frac{x_j}{\bar{d}_j(\mathbf{x})}\right| \leq \frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right).
\end{aligned} \tag{5.6}$$

We can derive that for one product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$:

$$\mathbb{P}\left(\left|\alpha_j x_j - \frac{x_j}{\bar{d}_j(\mathbf{x})}\right| \leq \frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right) = \mathbb{P}\left(\left|\alpha_j - \frac{1}{\bar{d}_j(\mathbf{x})}\right| \leq \frac{\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right) \tag{5.7a}$$

$$=\mathbb{P}\left(\left|\frac{\alpha_j - \frac{1}{d_j(\mathbf{x})}}{\alpha_j\frac{1}{\bar{d}_j(\mathbf{x})}}\right| \leq \frac{\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\alpha_j\frac{1}{\bar{d}_j(\mathbf{x})}\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}\right) = \mathbb{P}\left(\left|\bar{d}_j(\mathbf{x}) - \frac{1}{\alpha_j}\right| \leq \frac{\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\alpha_j\sqrt{n_j(\mathbf{x})}}\right). \tag{5.7b}$$

The exponential arrival time intervals are also sub-exponential random variables with parameters $(\frac{4}{\alpha_j^2}, \frac{2}{\alpha_j})$, denoted by SE$(\frac{4}{\alpha_j^2}, \frac{2}{\alpha_j})$, see Lemma B.2.1 in the appendix. Consider the concentration

inequality for i.i.d. exponential variables $\xi_i,\ i = 1, \dots n$ with mean $1/\lambda$:

$$\mathbb{P}\left[\left|\frac{1}{\lambda} - \frac{1}{n}\sum_{i=1}^{n}\xi_i\right| \geq t\right] \leq \begin{cases} 2\exp\left(-\frac{nt^2}{2\frac{4}{\lambda^2}}\right) & \text{if } 0 \leq t \leq \frac{2}{\lambda} \\ 2\exp\left(-\frac{nt}{2\frac{2}{\lambda}}\right) & \text{if } t > \frac{2}{\lambda} \end{cases}. \tag{5.8}$$

By the assumption $n_j(\mathbf{x}) \geq 8\log(|\mathcal{J}|^{\frac{1}{4}}T)$, we can compute

$$\left(\frac{\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\alpha_j\sqrt{n_m(\mathbf{x})}}\right)^2 = \frac{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}{\alpha_j^2 n_j(\mathbf{x})} \leq \frac{4}{\alpha_j^2},$$

and thus we can apply the first inequality in (5.8) to (5.7). Therefore, we can derive

$$\mathbb{P}\left(\left|\bar{d}_j(\mathbf{x}) - \frac{1}{\alpha_j}\right| \leq \frac{\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\alpha_j\sqrt{n_j(\mathbf{x})}}\right) \geq 1 - 2\exp\left(\frac{-n_j(\mathbf{x})\frac{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}{\alpha_j^2 n_j(\mathbf{x})}}{2\frac{4}{\alpha_j^2}}\right) \geq 1 - \frac{2}{|\mathcal{J}|\cdot T^4}$$

Then, we can compute a lower bound of (5.6) as

$$\left(1 - \frac{2}{|\mathcal{J}|\cdot T^4}\right)^{|\mathcal{J}_{\text{on}}(\mathbf{x})|} \geq \left(1 - \frac{2}{|\mathcal{J}|\cdot T^4}\right)^{|\mathcal{J}|} \geq 1 - \frac{2|\mathcal{J}|}{|\mathcal{J}|\cdot T^4} = 1 - \frac{2}{T^4}.$$

Q.E.D.

The Proposition 5.4.1 shows that after observing a certain amount of arrival time observations of turn-on products under a candidate price, we can derive a valid upper bound of the stead-state revenue rate with a high probability guarantee. Hence, we define the corresponding upper confidence bound and the lower confidence bound of the stationary revenue rate associated with price $\mathbf{x}$ by the end of batch m as follows.

Definition 5.4.1. *The upper confidence bound of the revenue rate associated with price* $\mathbf{x}$ *by the end of batch* m *is:*

$$U_m(\mathbf{x}) = \sum_{j\in\mathcal{J}_{on}(\mathbf{x})}\frac{x_j}{\bar{d}_j(\mathbf{x})} + \sum_{j\in\mathcal{J}_{on}(\mathbf{x})}\frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}, \tag{5.9}$$

and the lower confidence bound of the revenue rate of price $\mathbf{x}$ *by the end of batch* m *is:*

$$L_m(\mathbf{x}) = \sum_{j \in \mathcal{J}_{on}(\mathbf{x})} \frac{x_j}{\bar{d}_j(\mathbf{x})} - \sum_{j \in \mathcal{J}_{on}(\mathbf{x})} \frac{x_j \sqrt{32 \log(|\mathcal{J}|^{\frac{1}{4}} T)}}{\bar{d}_j(\mathbf{x}) \sqrt{n_j(\mathbf{x})}}. \tag{5.10}$$

Define $\mathbf{Rad}_m(\mathbf{x}) = \sum_{j \in \mathcal{J}_{\text{on}}(\mathbf{x})} \frac{x_j \sqrt{32 \log(|\mathcal{J}|^{\frac{1}{4}} T)}}{\bar{d}_j(\mathbf{x}) \sqrt{n_j(\mathbf{x})}}$ as the confidence radius of price $\mathbf{x}$ by end of batch m.

5.4.2 RNRM-UCB Algorithm

With the upper confidence defined in Definition 5.4.1, we can leverage the idea of UCB to develop an Online Batch Upper Confidence Bound Algorithm (RNRM-UCB) for our network revenue management problem. The UCB algorithm cannot be directly used in our setting because we cannot immediately gain or estimate the revenue that we shall collect after offering a price. One of the major difficulties of this problem is to quantify the loss of nonstationarity due to the transient performance of the Jackson Network when the price changes.

The key idea is to reduce the price changes and further reduce the intractable transient performance. We separate T periods into two phases. The first phase is the *Warm-up* Phase, where we offer each candidate price one by one and collect the system dynamic information. Consider offering price $\mathbf{x}$, we switch to a new price in next period once we collect $8 \log(|\mathcal{J}|^{1/4} T)$ number of arrival time intervals for each turn-on product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$ under the current price. Therefore, the total length of this Warm-up Phase is $O(|\mathcal{X}| \log(|\mathcal{J}|^{1/4} T))$. This Warm-up Phase is specifically designed to satisfy the requirement in Proposition 5.4.1. After this Warm-up Phase, the algorithm can eliminate the sub-exponential tail effects caused by the Poisson arrival process. Then, we can apply the concentration bound in 5.4.1, which is as tight as we can do for sub-Gaussian tails. The second phase is *Learning* Phase. We separate the Learning Phase into M number of batches. Define $I_m = 2^m$, $\forall m = 1, \ldots, M$. Each batch m contains $I_m L \tau$ periods, where $\tau = \log(T)(\log\log(T))^2$. Therefore, the batch size is exponentially increasing. We can then add batch m in the subscript of the following notations to represent their values by the end of batch m. We use $n_{mj}(\mathbf{x})$ to denote the number of arrived customers requesting product j under price $\mathbf{x}$. We use $\bar{d}_{mj}(\mathbf{x})$ to denote the empirical mean of the inter-arrival time, $\bar{d}_{mj}(\mathbf{x}) = \sum_{k=1}^{n_{mj}(\mathbf{x})} \hat{d}_{jk}(\mathbf{x}) / n_{mj}(\mathbf{x})$, for $j \in \mathcal{J}$.

At the beginning of each batch $m+1$, RNRM-UCB chooses price p with the highest $U_m(\mathbf{x})$ and offers this price during batch $m+1$. We present the algorithmic details in Algorithm 5.1.

Algorithm 5.1 Online Batch Upper Confidence Bound or Network Revenue Management with Reusable Resourses (RNRM-UCB).

Input: T, $\mathcal{X}$, service network, turn-off price.
Initialize: L, τ, I_m, M, $m = 0$.
Warm-up Phase:
for $\mathbf{x} \in \mathcal{X}$ **do**
  Offer $\mathbf{x}$, record inter-arrival time $\hat{d}_{jk}(\mathbf{x})$ for arriving customers of product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$.
  if $n_{mj}(\mathbf{x}) \geq 8\log(|\mathcal{J}|^{1/4}T)$ **then**
    Continue.
  end if
end for
Compute and update UCB for all prices as defined in (5.9).
Learning Phase:
for $m = 1, \ldots, M$ **do**
  Choose $\mathbf{x}_m = \text{argmax}_{\mathbf{x}\in} U_{m-1}(\mathbf{x})$.
  Offer $\mathbf{x}_m$ in batch m, i.e., for $I_m L\tau$ periods.
  Record inter-arrival time $\hat{d}_{jk}(\mathbf{x}_m)$ for arriving customers of product $j \in \mathcal{J}_{\text{on}}(\mathbf{x}_m)$.
  Compute and update UCB of $\mathbf{x}_m$ as defined in (5.9).
end for

5.4.3 RNRM-TS Algorithm

As introduced before, if the firm holds prior information about the rates associated with each price $\mathbf{x} \in \mathcal{X}$, then we can add the Bayesian assumption to this network revenue management problem, where the problem instance is drawn from a known prior distribution. By the conjugate prior theory (see, e.g., Raiffa and Schlaifer 1961, Diaconis and Ylvisaker 1979), we assume that RNRM-TS holds Gamma priors on $\alpha_j(\mathbf{x})$ of product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$ and the posteriors are updated by the end of batch m, $\forall m = 1, \ldots, M$. Specifically, we set the prior distribution of $\alpha_j(\mathbf{x})$ as **Gamma**$(\eta^{\alpha_j(\mathbf{x})}, \beta^{\alpha_j(\mathbf{x})})$. After observing one arrival time interval $\hat{d}_{jk}(\mathbf{x})$ of product j while offering price $\mathbf{x}$, the posterior distribution of $\alpha_j(\mathbf{x})$ is computed as **Gamma**$(\eta^{\alpha_j(\mathbf{x})} + 1, \beta^{\alpha_j(\mathbf{x})} + \hat{d}_{jk}(\mathbf{x}))$. To implement RNRM-TS, at the beginning of batch m, we sample parameters $\hat{\alpha}_j(\mathbf{x})$ for product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$, following the updated posterior distributions. Then we offer the price with the maximum sampled stationary network revenue rate $r(\hat{\alpha}_{\mathbf{x}}, \mathbf{x}) = \sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})} \mathbf{x}_j \hat{\alpha}_j(\mathbf{x})$. We present the algorithmic details in Algorithm 5.2.

5.5 Performance Analysis

In this section, we analyze the performance of the proposed two algorithms, RNRM-UCB and RNRM-TS, by bounding the relaxed regret defined in Definition 5.3.2. First, we derive the probability that (i) a network queueing (service) system reaches the steady state after a certain amount of

Algorithm 5.2 Online Batch Thompson Sampling for Network Revenue Management with Reusable Resources (RNRM-TS).

Input: T, $\mathcal{X}$, service network, turn-off price, $\eta^{\alpha_j(\mathbf{x})},\ \beta^{\alpha_j(\mathbf{x})},\ \forall j \in \mathcal{J}_{\text{on}}(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}$.
Initialize: L, τ, I_m, M, $m = 0$.
Warm-up Phase:
for $\mathbf{x} \in \mathcal{X}$ **do**
  Offer $\mathbf{x}$, record inter-arrival time $\hat{d}_{jk}(\mathbf{x})$ for arriving customers of product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$.
  if $n_{mj}(\mathbf{x}) \geq 8\log(|\mathcal{J}|^{1/4}T)$ **then**
    Continue.
  end if
end for
Update $\eta^{\alpha_j(\mathbf{x})},\ \beta^{\alpha_j(\mathbf{x})},\ \forall j \in \mathcal{J}_{\text{on}}(\mathbf{x}),\ \forall \mathbf{x} \in \mathcal{X}$.
Learning Phase:
for $m = 1, \ldots, M$ **do**
  Sample $\hat{\alpha}_j(\mathbf{x}) \sim$ **Gamma**$(\eta^{\alpha_j(\mathbf{x})},\ \beta^{\alpha_j(\mathbf{x})})$ for product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$, $\forall \mathbf{x} \in \mathcal{X}$
  Choose $\mathbf{x}_m = \text{argmax}_{\mathbf{x}_m \in \mathcal{X}} r(\hat{\alpha}_{\mathbf{x}}, \mathbf{x})$.
  Offer $\mathbf{x}_m$ in batch m, i.e., for $I_m L\tau$ periods.
  Record inter-arrival time $\hat{d}_{jk}(\mathbf{x}_m)$ for arriving customers of product $j \in \mathcal{J}_{\text{on}}(\mathbf{x}_m)$.
  Compute and update $\eta^{\alpha_j(\mathbf{x}_m)},\ \beta^{\alpha_j(\mathbf{x}_m)},\ \forall j \in \mathcal{J}_{\text{on}}(\mathbf{x}_m)$.
end for

time starting from an empty state (i.e., zero customers in the system) under a fixed price (§5.5.1.1) and (ii) a system reaches the steady state after a certain amount of time starting from the steady state under a previously implemented price (§5.5.1.2). Then, based on the probability analysis, we provide the regret bound for RNRM-UCB in §5.5.2 and for RNRM-TS in §5.5.3. A high-level roadmap of our analysis is shown in Figure 5.2.

5.5.1 High Probability Bound for Mixing Time

In a network service system defined in Section 5.2, the collected revenue is affected by both the demand rates and the resource utilization. Recall that the service system can be reduced to a Jackson network under a specific offered price vector, upon completion of existing customers under the previously posted price. However, the resource utilization during the transient state of a Jackson network is intractable and impossible to analyze. Therefore, in this section we evaluate how quickly a Jackson network can reach the steady state.

When the firm starts service from the empty state, i.e., zero customers in the network service system, or switches to another price from the currently offered price, the service (queueing network) system enters a transient state and needs a certain amount of time to reach the steady state again. To compute the expected revenue collected during this time, referred to as the transient state, is intractable. Therefore, the time spent on reaching the steady state, which is also referred to as the

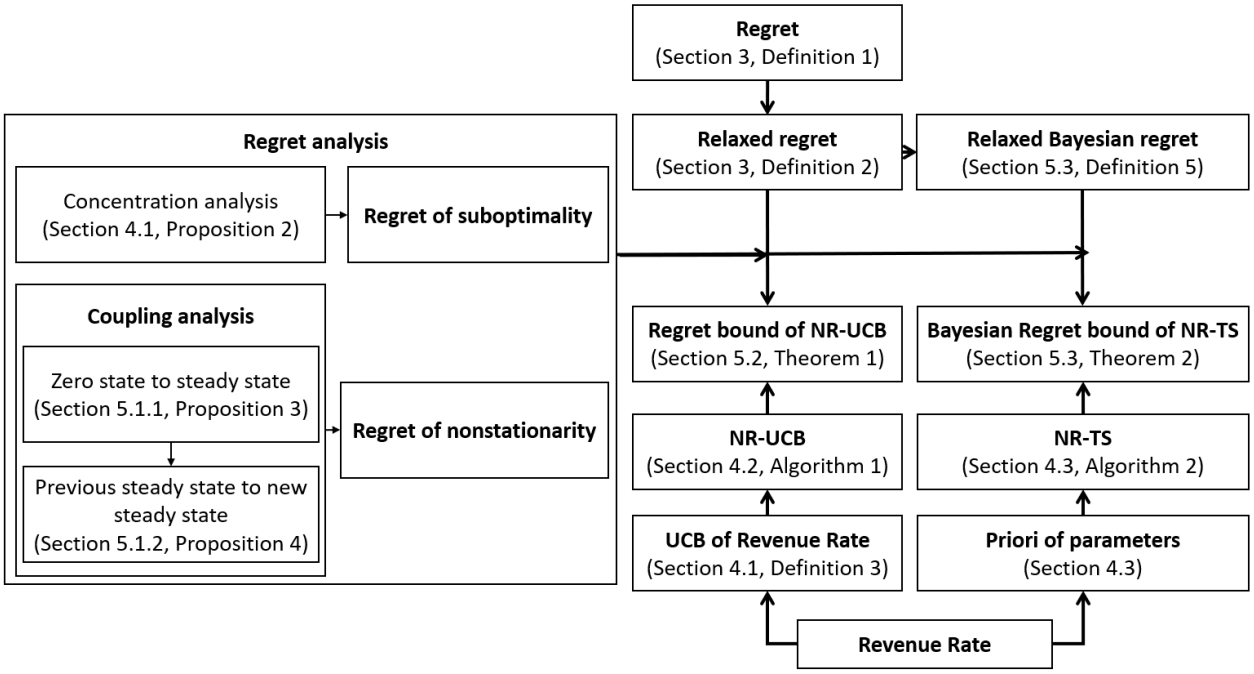


Figure 5.2: Roadmap of RNRM-UCB and RNRM-TS Algorithms and Regret Analysis.

mixing time, is crucial for computing the practical revenue under a specific policy and thus crucial for the regret analysis. To the best of our knowledge, this is the first result giving a finite-time high probability bound on mixing times of Jackson Network system (upon action changes).

In this section, we start from some properties of Jackson Network and then derive the probabilistic bound for the system when starting from from an empty state (§5.5.1.1) and the steady state under a previously implemented price (§5.5.1.2). Technically, we develop a new coupling argument between this target system and a virtual system starting from a state sampled from the steady-state distribution and maintaining the steady state thereafter.

Poisson Arrival and Departure. Under the steady-state of an M/M/c queue (if the steady-state exists, i.e., $\rho = \lambda/c\mu < 1$), then the process following which the customers leaving the system is also a Poisson process with rate λ. Therefore, one can easily derive that the arrival as well as the departure processes of each type of server (each node in the network representation) are Poisson processes with the same rate λ_i under the steady-state, given the utilization factor of the system is less then 1, i.e., the utilization factors of all nodes are less than 1 (see the famous Burke's Theorem Burke 1956).

Our Chapter 4 is the only work for studying the mixing time for an M/M/c queue. However, this results cannot be directly applied in analyzing the mixing time for a Jackson network. First of all, this work requires that the arrival process is a well-defined Poisson process. For a Jackson network, the arrival process of each queue during the transient process is not Poisson any more. To achieve

a well-defined Poisson arrival process, one intuition is to let the queues, corresponding to each resource, to reach the steady-state sequentially following the topological order (see Assumption 5.2.1). Thus, the arrival processes of the later queues are well-defined Poisson processes conditional on that the previous queues have all already reached the steady state. Then the problem is the starting points of the later queues. Chapter 4 considers stating from zero state or a steady state over another set of system parameter. It is obviously that the starting point of later queues are not empty state nor a steady state under another set of parameters, and these two starting points cannot be achieved by rejecting customers. Even this starting point problem is solved, this approach induces a high-probability mixing time that is linear on the number of resources. This result is not desired for systems with multiple, especially large number of resources.

In this section, we develop new mixing time analysis for Jackson network. On the one hand, we construct a virtual queue and an alternative queue to help understand the system dynamics of the later queues after the Jackson network changes. On the other hand, we define node layers for Jackson network, which form the basis of our mixing time analysis. We let the nodes in the same layer reach steady state by a same time and analyze the layers one by one. In short, we derive a high-probability mixing time that is linear on the number of layers of a network, which is typically extremely less than the number of resources.

Definition 5.5.1. *We divide the nodes (types of servers) into layers indexed by the maximum number of previously occupied nodes (including the virtual source node) along the service routes of all products. For example, Nodes 1 and 2 of the illustrative example (see Figure 5.1) is Layer-1 nodes and Node 3 is a Layer-2 node. We denote the number of layers by L and the set of Layer-l node as $\mathcal{I}_l$.*

In the following analysis, we will utilize the aforementioned Poisson arrival and departure property of the nodes in a Jackson Network to bound the mixing time of the nodes layer by layer.

5.5.1.1 Coupling Analysis for Jackson Network Starting from Empty State.

We consider a Jackson Network starting from empty state with a fixed set of system parameters, i.e., $\lambda_i,\ \mu_i$ for $i \in \mathcal{I}$. We define events as follows (all starting from empty state).

$$\mathcal{A}_t = \{\text{the Jackson Network reaches the steady-state by time } t\},$$

$$\mathcal{A}_{t,l} = \{\text{all layer-l nodes reach the steady-state by time } t\},$$

$$\mathcal{A}_t^i = \{\text{the node } i \text{ reaches the steady-state by time } t\}.$$

Proposition 5.5.1. ***(Coupling Probability of a Jackson Network starting from Empty State)*** *For $t \geq L\tau$, where $\tau = \log(T)(\log\log(T))^2$, the probability that a Jackson Network reaches the*

steady-state is bounded as follows.

$$\mathbb{P}(\mathcal{A}_{L\tau}) \geq 1 - \frac{2|\mathcal{I}|}{T^3}.$$

Proof of Proposition 5.5.1. In this proof, we analyze the coupling time of nodes layer by layer. First, we discuss the coupling analysis of an M/M/c queue, which is the basis for completing this proof. With the assumption $\log\log(T) > 3$, the probability that an M/M/c queue reaches the steady state from empty state after $\log(T)(\log\log(T))^2$ periods of time is bounded by $1 - 2/T^3$, see Proposition 4.4.1 in Chapter 4. This results are derived following the law of total probability, the first-order stochastic dominance between busy times of an M/M/c queue, and the concentration inequality for independent samples. We denote node i is a prerequisite node of i', $i < i'$ if there exists a product requires to use nodes i and i'. Thus, it is easy to show that the prerequisite nodes of a Layer-l node must have smaller layer indices.

In our Jackson Network, node i' reaches its steady-state indicates that the arrival process of node i' is a stable Poisson process, and therefore it requires that all the prerequisite nodes also reach the steady-state. In this proof, we analyze the nodes layer by layer. By treating the nodes in the same layer together, we can derive that

$$\begin{aligned}
\mathbb{P}(\mathcal{A}_{L\tau}) &= \mathbb{P}(\mathcal{A}_{L\tau,1}\mathcal{A}_{L\tau,2}\mathcal{A}_{L\tau,3}\ldots\mathcal{A}_{L\tau,L}) \\
&\geq \mathbb{P}(\mathcal{A}_{\tau,1}\mathcal{A}_{2\tau,2}\mathcal{A}_{3\tau,3}\ldots\mathcal{A}_{L\tau,L}) \\
&= \mathbb{P}(\mathcal{A}_{\tau,1})\mathbb{P}(\mathcal{A}_{2\tau,2}\mathcal{A}_{3\tau,3}\ldots\mathcal{A}_{L\tau,L}|\mathcal{A}_{\tau,1}) \\
&= \mathbb{P}(\mathcal{A}_{\tau,1})\mathbb{P}(\mathcal{A}_{2\tau,2}|\mathcal{A}_{\tau,1})\mathbb{P}(\mathcal{A}_{3\tau,3}\ldots\mathcal{A}_{L\tau,L}|\mathcal{A}_{\tau,1}\mathcal{A}_{2\tau,2}) \\
&= \mathbb{P}(\mathcal{A}_{\tau,1})\prod_{l=2}^{L}\mathbb{P}(\mathcal{A}_{l\tau,l}|\mathcal{A}_{\tau,1}\ldots\mathcal{A}_{(l-1)\tau,l-1}).
\end{aligned} \tag{5.11}$$

In short, we derive a lower bound of this mixing probability by requiring the Layer-l nodes reach the steady-state within the first $L \times \tau$ periods, conditional on that Layer-l' nodes also reach the steady-state within the first $l' \times \tau$ periods for $l' < l$, or equivalently, all prerequisite nodes of Layer-l nodes reach the steady state within $(l-1) \times \tau$ periods.

Layer-1 Nodes. In the first τ periods, by applying the probability bound for an M/M/c queue, we can easily conclude that all Layer-1 nodes (the correspondent M/M/c queues) reach the steady-state with probability $1 - \frac{2}{T^3}$. Recall $\mathcal{I}_1$ is the collection of all Layer-1 nodes and $|\mathcal{I}_1|$ is the cardinality. Therefore we can compute that the probability that all Layer-1 nodes reach the steady-

state by first τ periods is

$$\mathbb{P}(\mathcal{A}_{\tau,1}) = \left(1 - \frac{2}{T^3}\right)^{|\mathcal{I}_1|} \geq 1 - \frac{2|\mathcal{I}_1|}{T^3}, \tag{5.12}$$

because the queueing evolution of all Layer-1 nodes are independent with each other and thus we can simplify multiply the probability together, in the first equality in the above expression.

Layer-2 Nodes. We then analyze the Layer-2 nodes. Consider one arbitrary layer-2 node, denoted by i without loss of generality. We then focus on the corresponding queue of this node i conditional on that all Layer-1 nodes have reached steady-state. We firstly show a virtual queue can reach steady state within the next τ periods with probability at least $1 - \frac{2}{T^3}$. Then we show that an alternative queue of the actual queue takes a shorter time to reach steady state and thus the actual queue can also reach steady state within the next τ periods with probability at least $1 - \frac{2}{T^3}$.

- *Virtual queue.* We consider a virtual queue which rejects all customers arrived before τ periods. Then by Proposition 4.4.1, this virtual queue reaches the steady state with probability $1 - \frac{2}{T^3}$ after the next τ periods, i.e., after the first 2τ periods.

- *Alternative queue.* Denote the probability that the number of customers in Node i at the end of period τ is less than N by $\mathbb{P}_i(N)$, including the customers both using the server or in the waiting queue of node i. The actual queue accepts customers during the first τ periods with a changing rate from an initial rate to the stable arrival rate λ_i (conditional on all Layer-1 nodes reach steady state and thus provides stable Poisson arrivals for nodes with high layer indices). Denote the demand of all products to the Jackson Network by α. The stable arrival rate $\lambda_i = \alpha p_{si} + \alpha \sum_{i':(i',i\in E)} p_{si'} p_{i'i}$, where αp_{si} is the initial rate, i.e., the demand rate of products that request node i as the first service, and $\alpha p_{si'} p_{i'i}$ is the stable demand flowing from prerequisite node i' to node i. Consider an alternative queue which accepts customers at the stable rate λ_i. If the queue starts from time 0, then the probability that this virtual queue has customers with a size less than N at the end of period τ is smaller than $\mathbb{P}_i(N)$. This is because this alternative queue accepts more customers while the service times are all following a same distribution. Then we postpone the starting point of this alternative queue to a time stamp between 0 and τ such that the probability that this alternative queue has customers with a size less than N at the end of period τ is the same as $\mathbb{P}_i(N)$. Note that this time stamps is exist, which can be easily proved by the memoryless property of the exponential service time assumptions. Then we have the following two results. First, this alternative queue has a same evolution path as the actual queue. Second, because this alternative queue starts accepting customers earlier than time τ, and thus it will reach steady state earlier than the virtual queue.

We provide an illustration of the actual, alternative, and virtual queues in Figure 5.3.

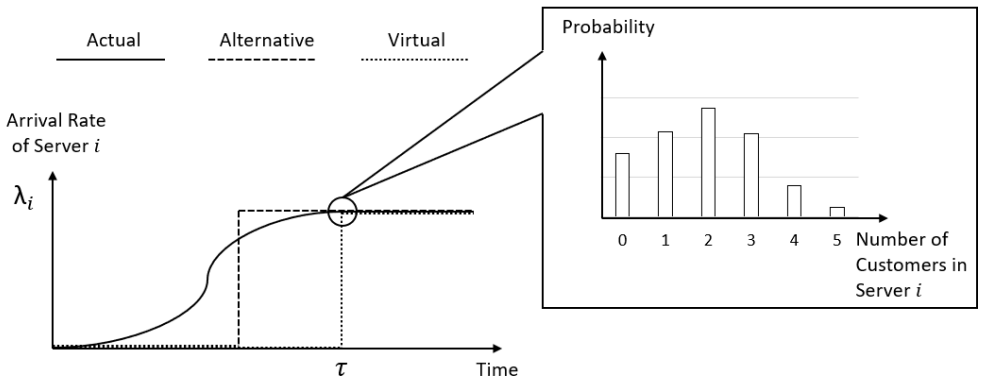


Figure 5.3: The left part shows the arrival rates of the actual, alternative, and virtual queues (for a Layer-2 node i when the Jackson network starting from empty state). The right part shows the probability that the queue has different number of customers at time τ. This probability are the same for the alternative and the actual queues and thus they are equivalent systems after time τ.

To conclude, we have the actual queue can reach the steady-state with a higher probability after the first 2τ periods. Similarly, the queueing evolution of all Layer-2 nodes are independent with each other and thus we can simplify multiply the probability together. Therefore, we have that for all Layer-2 nodes

$$\mathbb{P}(\mathcal{A}_{2\tau,2}|\mathcal{A}_{\tau,1}) = \left(1 - \frac{2}{T^3}\right)^{|\mathcal{I}_2|} \geq 1 - \frac{2|\mathcal{I}_2|}{T^3}. \tag{5.13}$$

Layers with Higher Indices. Then we apply the same analysis steps of the Layer-2 nodes to nodes in layers with higher indices. Conditional on the events that all nodes in layers with lower indices have reached the steady state by the end of $(l-1)\tau$ periods, we can compute that the probability the all nodes of layer l can reach steady state in the next τ period with a probability:

$$\mathbb{P}(\mathcal{A}_{l\tau,l}|\mathcal{A}_{\tau,1}\ldots\mathcal{A}_{(l-1)\tau,l-1}) \geq 1 - \frac{2|\mathcal{I}_l|}{T^3}. \tag{5.14}$$

With the results in (5.12), (5.13), and (5.14), we are ready to bound the probability in (5.11):

$$\mathbb{P}(\mathcal{A}_{L\tau}) \geq \prod_{l=1}^{L}\left(1 - \frac{2|\mathcal{I}_l|}{T^3}\right) \geq 1 - \frac{2|\mathcal{I}|}{T^3}.$$

We refer to Figure 5.4 for a direct representation of the above proof sketch, which shows the mixing time analysis of an illustrative example.

Q.E.D.

Proposition 5.5.1 says that it is more difficult for a Jackson Network to reach the steady state if

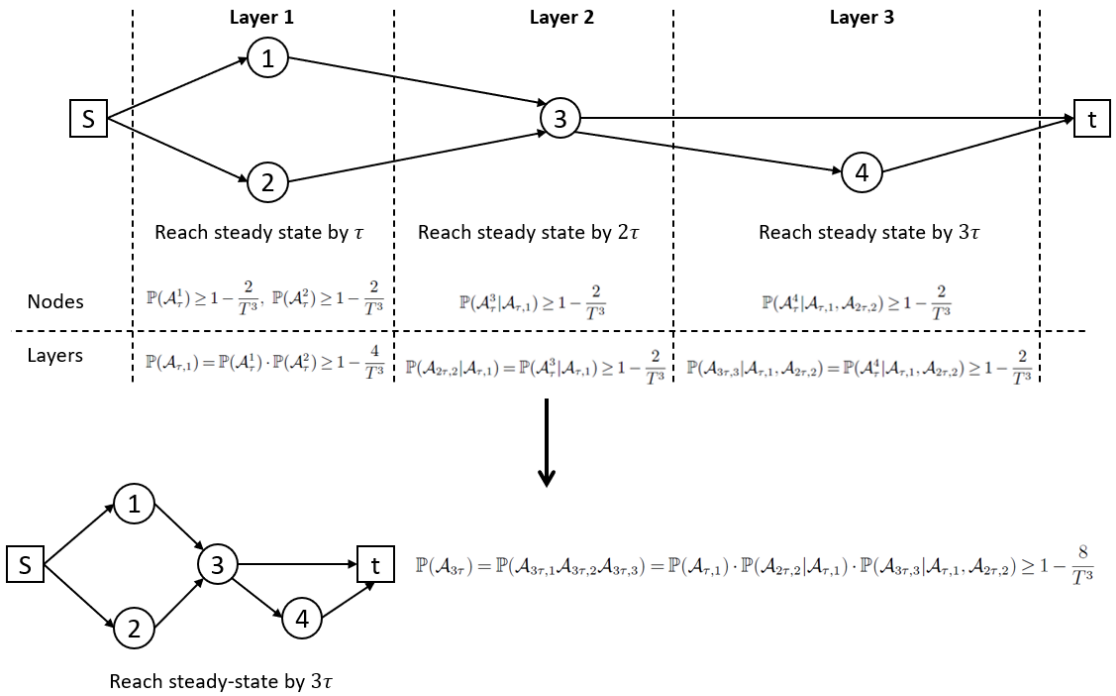


Figure 5.4: The mixing time analysis for an illustrative example starting from zero state.

it has a larger size. From one hand, a Jackson Network with more node layers will take a longer time to reach the steady state. From the other hand, the probabilistic bound for the mixing time could be looser for a Jackson Network with more nodes while keeping the order of the bound as the same.

5.5.1.2 Coupling Analysis for Jackson Network when Price Changes.

With the results in Proposition 5.5.1, we can bound the mixing time for the Jackson Network in the beginning of implementing a pricing policy, i.e., after posting the first price and the service system runs from an empty state. However, our proposed pricing policies under this incomplete information scenario switches between candidate prices and our batch setting ensures that the operation of the Jackson Network is in the steady state before switching to another candidnate price. Therefore, the transient state of the Jackson Network, as well as each queue (or each node), can be induced not only by starting from empty state but also by starting from a steady state of a previously posted price. It is possible that when a new price is posted, the customers arrived under the previously posted price still remain in the system. In order to compute the expected collected revenue when implementation the policy, we also need to ensure that the coupling occurs with high probability even with price changes (with the aid of Assumption 5.2.5), which is encapsulated in Proposition 5.5.2.

We similarly define the following events when the Jackson Network starts to operate under a new price starting from the steady state of the old price:

$$\mathcal{B}_t = \{\text{the Jackson Network reaches the steady-state by time } t\},$$

$$\mathcal{B}_{t,l} = \{\text{all layer-l nodes reach the steady-state by time } t\},$$

$$\mathcal{B}_t^i = \{\text{the node } i \text{ reaches the steady-state by time } t\}.$$

Proposition 5.5.2. ***(Coupling Probability of Jackson Network starting from Steady-state of One Candidate Price.)*** *For $t \geq 2L\tau$, where $\tau = \log(T)(\log\log(T))^2$, the probability that the Jackson Network reaches the steady-state is bounded as follows.*

$$\mathbb{P}(\mathcal{B}_{2L\tau}) \geq 1 - \frac{4|\mathcal{I}|}{T^3}.$$

Proof of Proposition 5.5.2. Similar to the proof of Proposition 5.5.2, we analyze the mixing time of nodes layer by layer. Before conducting the proof, we introduce an existing work for an M/M/c queue mixing time. Proposition 4.4.2 in Chapter 4 says that for one M/M/c queue where Assumption 5.2.5 holds, then the probability that the queue reaches a new steady state from a old steady state within 2τ time periods after the price change is bounded by $1 - 4/T^3$. To derive this result, the authors decompose possible price changes into the following two cases. The first is when the arrival rate changes from a higher value to a lower value and the second is vice versa. For each case, the authors further consider two sub-cases: the unit service rate increase or decrease.

Consider the Jackson Network system in our problem. We can bound the probability by:

$$\mathbb{P}(\mathcal{B}_{2L\tau}) \geq \mathbb{P}(\mathcal{B}_{2\tau,1}) \prod_{l=2}^{L} \mathbb{P}(\mathcal{B}_{2l\tau,l} | \mathcal{B}_{2\tau,1} \ldots \mathcal{B}_{2(l-1)\tau,l-1}). \tag{5.15}$$

The above inequality is derived following the same steps in (5.11) and we neglect the details here.

Assumption 5.2.5 allows us to apply Proposition 4.4.2 when we analyze one specific node, i.e., one specific M/M/c queue. The structure of the following analysis is similar to the structure of the proof of Proposition 5.5.1, we first analyze Layer-1 node and then the nodes with higher layer indices. For nodes with higher layer indices, we apply the aforementioned proposition to a virtual queue, and then show this virtual queue takes longer than an alternative system with the actual queue.

Layer-1 Nodes. In the first 2τ periods, by applying the probability bound for an M/M/c queue with price change (Proposition 4.4.2), we can easily conclude that all Layer-1 nodes (the correspondent M/M/c queues) reach the steady-state with probability $1 - \frac{4}{T^3}$. Consider the probabilistic

independency of the coupling process among Layer-1 nodes, the probability that all Layer-1 nodes reach the steady-state by first 2τ periods is

$$\mathbb{P}(\mathcal{B}_{2\tau,1}) = \left(1 - \frac{4}{T^3}\right)^{|\mathcal{I}_1|} \geq 1 - \frac{4|\mathcal{I}_1|}{T^3}. \tag{5.16}$$

Layer-2 Nodes and Nodes with Higher Layer Indices. We then analyze nodes with a layer index 2 or higher. Consider one arbitrary Layer-l node, denoted by i without loss of generality. We then focus on the corresponding queue of this node i conditional on that all prerequisites nodes have reached steady-state. We firstly show a virtual queue can reach steady state within the next 2τ periods with probability at least $1 - \frac{4}{T^3}$. Then we show that an alternative queue of the actual queue takes a shorter time to reach steady state and thus the actual queue can also reach steady state within the next 2τ periods with probability at least $1 - \frac{4}{T^3}$.

- *Virtual queue.* We consider a virtual queue which accepts customers under the old queueing parameters in the first $2(l-1)\tau$ periods and then accepts customers under the new queueing parameter in the next 2τ periods. With Assumption 5.2.5, we can directly apply the results for one M/M/c queue to reach steady state starting from a steady state and conclude that this virtual queue reaches the steady state with probability $1 - \frac{4}{T^3}$ after the next 2τ periods, i.e., after the first $2l\tau$ periods.

- *Alternative queue.* Denote the probability that in Node i at the end of $2(l-1)\tau$ periods, the number of old customers is less than N_o and the number of new customers is less than N_n by $\mathbb{P}_i(N_o, N_n)$, including the customers both using the server or in the waiting queue of node i. Consider an alternative queue which starts accepting new customers under the new queueing system parameter from time 0 and denote the corresponding probability for having less than N_o number of old customers and less than N_n new customers at the end of $2(l-1)\tau$ periods by $\mathbb{P}'_i(N_o, N_n)$. By the analysis in Proposition 5.5.1, we know there is a mismatch between $\mathbb{P}_i(N_o, N_n)$. Then we postpone the starting point of this alternative queue to a time stamp between 0 and $2(L-1)\tau$ such that $\mathbb{P}'_i(N_o, N_n) = \mathbb{P}_i(N_o, N_n)$. Note that this time stamps is always exist, which can be easily proved by the memoryless property of the exponential service time assumptions because we can always find this stamp backwards from any customer state at time $2(l-1)\tau$. Then we have the following two results. First, this alternative queue has a same evolution path as the actual queue. Second, because this alternative queue starts accepting new customers earlier than time $2(l-1)\tau$, and thus it will reach steady state earlier than the virtual queue.

To conclude, we have the actual queue can reach the steady-state with a higher probability after the first 2τ periods. Therefore , conditional on the events that all nodes in layers with lower indices

have reached the steady state by the end of $2(l-1)\tau$ periods, we can compute that the probability the all nodes of layer l can reach steady state in the next 2τ period with a probability:

$$\mathbb{P}(\mathcal{B}_{2l\tau,l}|\mathcal{B}_{2\tau,1}\dots\mathcal{B}_{2(l-1)\tau,l-1}) \geq 1 - \frac{4|\mathcal{I}_l|}{T^3}. \tag{5.17}$$

With the results in (5.16) and (5.17), we are ready to bound the probability in (5.15):

$$\mathbb{P}(\mathcal{B}_{2L\tau}) \geq \prod_{l=1}^{L}\left(1 - \frac{4|\mathcal{I}_l|}{T^3}\right) \geq 1 - \frac{4|\mathcal{I}|}{T^3}.$$

Q.E.D.

Proposition 5.5.2 reveals same insights as that of Proposition 5.5.1. It indicates that it is more difficult for a Jackson Network to reach a new steady state when the system dynamic changes if it has a larger size. From one hand, a Jackson Network with more node layers will take a longer time to reach a new steady state. From the other hand, the probabilistic bound for the mixing time could be looser for a Jackson Network with more nodes while keeping the order of the bound as the same.

5.5.2 Performance of RNRM-UCB

In this section, we provide a theoretical probabilistic regret bound for the RNRM-UCB algorithm. We bound the relaxed regret, which is defined in Definition 5.3.2 and is a valid upper bound of the regret. The high probability bound for mixing time of Jackson Network developed in §5.5.1 help bound the loss due to the nonstationarity, and thus we are ready to carry out the regret analysis of RNRM-UCB.

Theorem 5.5.1. *The T-period cumulative regret of RNRM-UCB is bounded by* $O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\log T\log(|\mathcal{J}|^{1/4}T)}\right)$.

Note that the regret lower bound for nominal MAB is $\Omega(\sqrt{|\mathcal{J}||\mathcal{X}|T})$ (see, e.g., Theorem 2.1 in Slivkins et al. (2019)), which suggests that our regret bound is tight, up to a logarithmic factor.

Proof of Theorem 5.5.1. We bound the relaxed regret during the Warm-up and Learning Phases separately. We further decompose the relaxed regret during the Learning Phase into two parts. The first part is the regret of suboptimality, which is induced by not offering the state-independent optimal price in certain time periods. We bound the loss due to suboptimality with the help of Proposition 5.4.1. The second part is the regret of nonstationarity, which represents the revenue loss induced by changing offered prices. We bound the loss due to nonstationarity with the help of Propositions 5.5.1 and 5.5.2. The structure of this proof is similar to that of Theorem 1 in Jia et al. (2020b), where the authors consider a single reusable product setting.

Regret of the Warm-up Phase. The difference between $J^{\text{LP}}_{\text{Warm-up}}$ and $J^{\text{RNRM-UCB}}_{\text{Warm-up}}$ is at most linear on the length of the Warm-up Phase, which is $O(P\log(T))$. Therefore we can directly have

$$\overline{\text{Regret}}_{\text{Warm-up}}(\text{RNRM-UCB}, T) = O(|\mathcal{X}|\log(T)). \tag{5.18}$$

Regret of nonstationarity during the Learning Phase. We denote the offered price of batch m in RNRM-UCB as $\mathbf{x}_m$. Therefore, the expected revenue collected in batch m is

$$J^{\text{RNRM-UCB}}_m(\mathbf{x}_m) = J^{\text{RNRM-UCB}}_{m,0\sim 2L\tau}(\mathbf{x}_m) + J^{\text{RNRM-UCB}}_{m,2L\tau\sim I_m L\tau}(\mathbf{x}_m),$$

where $J^{\text{RNRM-UCB}}_{m,0\sim 2L\tau}(\mathbf{x}_m)$ denotes the expected revenue collected in the first $2L\tau$ periods and $J^{\text{RNRM-UCB}}_{m,2L\tau\sim I_m L\tau}(\mathbf{x}_m)$ denotes the expected revenue collected from $(2L\tau+1)$th period to the end of batch m. We define event $\mathcal{C}_{2L\tau}$ as that the network service system reaches the steady state under the new price within the first $2L\tau$ periods and define $\bar{\mathcal{C}}_{2L\tau}$ as the complementary event of $\mathcal{C}_{2L\tau}$. By Propositions 5.5.1 and 5.5.2, we have $\mathbb{P}[\mathcal{C}_{2L\tau}] \geq 1 - 4|\mathcal{I}|/T^3$. Consequently, we define $J^{\text{RNRM-UCB}}_{m,\mathcal{C}_{2L\tau}}(\mathbf{x}_m)$ to be the expected value of $J^{\text{RNRM-UCB}}_{m,2L\tau\sim I_m L\tau}(\mathbf{x}_m)$ conditional on $\mathcal{C}_{2L\tau}$ and $J^{\text{RNRM-UCB}}_{m,\bar{\mathcal{C}}_{2L\tau}}(\mathbf{x}_m)$ similarly. As a result, we can express the expected revenue during batch m as:

$$\begin{aligned}
J^{\text{RNRM-UCB}}_m(\mathbf{x}_m) &= J^{\text{RNRM-UCB}}_{m,0\sim 2L\tau}(\mathbf{x}_m) + J^{\text{RNRM-UCB}}_{m,\mathcal{C}_{2L\tau}}(\mathbf{x}_m)\mathbb{P}(\mathcal{C}_{2L\tau}) + J^{\text{RNRM-UCB}}_{m,\bar{\mathcal{C}}_{2L\tau}}(\mathbf{x}_m)\big(1-\mathbb{P}(\mathcal{C}_{2L\tau})\big) \\
&\geq O(\tau) + \left(\vec{\alpha}^T_{\mathbf{x}_m}\mathbf{x}_m\right)(I_m L\tau - 2L\tau)\left(1 - \frac{4|\mathcal{I}|}{T^3}\right) + O\left(\frac{(I_m L\tau - 2L\tau)|\mathcal{I}|}{T^3}\right) \\
&= \left(\vec{\alpha}^T_{\mathbf{x}_m}\mathbf{x}_m\right) I_m L\tau - O(\tau) \\
&= r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m) I_m L\tau - O(\tau).
\end{aligned} \tag{5.19}$$

Regret of suboptimality during the Learning Phase. Recall the upper and lower confidence bounds of the revenue rate defined in Definition 5.4.1. It is clear that

$$U_m(\mathbf{x}) - L_m(\mathbf{x}) = 2\mathbf{Rad}_m(\mathbf{x}) = 2\sum_{j\in\mathcal{J}_{\text{on}}(\mathbf{x})} \frac{x_j\sqrt{32\log(|\mathcal{J}|^{\frac{1}{4}}T)}}{\bar{d}_j(\mathbf{x})\sqrt{n_j(\mathbf{x})}}.$$

For batch m, we can compute the difference in the revenue rates between the optimal price and selected price $\mathbf{x}_m$ as

$$\Delta(\mathbf{x}_m) = r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m) \tag{5.20a}$$

$$= \big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\mathbf{x}_m)\big) + \big(U_m(\mathbf{x}_m) - L_m(\mathbf{x}_m)\big) + \big(L_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m)\big) \tag{5.20b}$$

$$\leq \big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big) + \big(U_m(\mathbf{x}_m) - L_m(\mathbf{x}_m)\big) + \big(L_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m)\big), \tag{5.20c}$$

where the last inequality holds because $U_m(\mathbf{x}_m) \geq U_m(\tilde{\mathbf{x}})$ by the algorithm construction.

Then the cumulative regret during the Learning Phase is

$$\begin{aligned}
\overline{\text{Regret}}_{\text{Learning}}(\text{RNRM-UCB}, T) &= J^{\text{LP}}_{\text{Learning}} - J^{\text{RNRM-UCB}}_{\text{Learning}} && (5.21a)\\
&= \sum_{m=1}^{M} \left(r\left(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}\right) I_m L\tau - J_m^{\text{RNRM-UCB}}\left(\mathbf{x}_m\right) \right) && (5.21b)\\
&= \sum_{m=1}^{M} \mathbb{E}\Big[\Delta(\mathbf{x}_m) I_m L\tau + O(\tau)\Big]. && (5.21c)
\end{aligned}$$

By (5.20), we have

$$\begin{aligned}
\sum_{m=1}^{M} \mathbb{E}\Big[\Delta(\mathbf{x}_m) I_m L\tau\Big] &= \sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big)\Big] I_m L\tau + \sum_{m=1}^{M} \mathbb{E}\Big[\big(U_m(\mathbf{x}_m) - L_m(\mathbf{x}_m)\big)\Big] I_m L\tau\\
&\quad + \sum_{m=1}^{M} \mathbb{E}\Big[\big(L_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m)\big)\Big] I_m L\tau.
\end{aligned} \tag{5.22}$$

We first analyze the middle term of (5.22). Let $\mathcal{M}_{\mathbf{x}}$ denote the batches choosing price $\mathbf{x}$ and $T_{\mathbf{x}}$ the number of periods choosing price $\mathbf{x}$, i.e., $T_{\mathbf{x}} = \sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau$. Then,

$$\begin{aligned}
\sum_{m=1}^{M} \mathbb{E}\Big[\big(U_m(\mathbf{x}_m) - L_m(\mathbf{x}_m)\big) I_m L\tau\Big] &= \sum_{\mathbf{x} \in \mathcal{X}} \sum_{m \in \mathcal{M}_{\mathbf{x}}} \mathbb{E}\big[U_m(\mathbf{x}) - L_m(\mathbf{x})\big] I_m L\tau && (5.23a)\\
&= \sum_{\mathbf{x} \in \mathcal{X}} \sum_{m \in \mathcal{M}_{\mathbf{x}}} 2\mathbb{E}\big[\mathbf{Rad}_m(\mathbf{x})\big] I_m L\tau && (5.23b)\\
&\leq \sum_{\mathbf{x} \in \mathcal{X}} O\left(|\mathcal{J}| \sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}} \log(|\mathcal{J}|^{1/4} T)} \right) && (5.23c)\\
&= O\left(|\mathcal{J}| \sqrt{\log(|\mathcal{J}|^{1/4} T)} \right) \sum_{\mathbf{x} \in \mathcal{X}} \sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}}} && (5.23d)\\
&\leq O\left(|\mathcal{J}| \sqrt{\log(|\mathcal{J}|^{1/4} T)} \sqrt{\log T} \right) \sum_{\mathbf{x} \in \mathcal{X}} \sqrt{T_{\mathbf{x}}} && (5.23e)\\
&\leq O\left(\sqrt{|\mathcal{J}||\mathcal{X}| T \log T \log(|\mathcal{J}|^{1/4} T)} \right), && (5.23f)
\end{aligned}$$

where (5.23c) by Lemma 5.5.1, (5.23e) is derived by the number of the batches, and (5.23f) follows Jensen's inequality. The number of batches is $\log_2\left(1 + \frac{T - 8|\mathcal{X}| \log(|\mathcal{J}|^{1/4} T)}{2L \log(T^2)(\log\log(T))^3}\right) = O\big(\log(T)\big)$.

Below we analyze the first term of (5.22), and the same logic applies to the third term, of which

we omit the details here.

$$\sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big)\Big] I_m L\tau \tag{5.24a}$$

$$\leq \sum_{m=1}^{M} \mathbb{E}\Big[\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big)\mathbb{1}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big)\Big] I_m L\tau \tag{5.24b}$$

$$\leq \sum_{m=1}^{M} \mathbb{E}\Big[r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}})\mathbb{1}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big) - U_m(\tilde{\mathbf{x}})\mathbb{1}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big)\Big] I_m L\tau \tag{5.24c}$$

$$= \sum_{m=1}^{M} r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}})\mathbb{P}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big) I_m L\tau - \sum_{m=1}^{M} \mathbb{E}\Big[U_m(\tilde{\mathbf{x}})\mathbb{1}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big)\Big] I_m L\tau \tag{5.24d}$$

$$\leq \sum_{m=1}^{M} r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}})\mathbb{P}\big(r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) > U_m(\tilde{\mathbf{x}})\big) I_m L\tau \tag{5.24e}$$

$$\leq O\left(\frac{1}{T^3}\right), \tag{5.24f}$$

where (5.24e) is because $U_m(\tilde{\mathbf{x}}) > 0$ by Definition 5.4.1, and (5.24f) follows Proposition 5.4.1.

Given that $O\big(n_{Mj}(\mathbf{x})\big) = O(T_{\mathbf{x}})$, for any product $j \in \mathcal{J}_{\text{on}}(\mathbf{x})$, we have $\sum_{m=1}^{M} O(\tau) = O\big((\log(T))^2(\log\log(T))^2\big)$. Combining the three terms together,

$$\begin{aligned}
\overline{\text{Regret}}_{\text{Learning}}(\text{RNRM-UCB}, T) =& \sum_{m=1}^{M} \mathbb{E}\Big[\Delta(\mathbf{x}_m) I_m L\tau\Big] + \sum_{m=1}^{M} O(\tau) \\
=& O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\sqrt{\log T}\log(|\mathcal{J}|^{1/4}T)}\right) + O\left(\frac{1}{T^3}\right) \\
&+ O\big((\log(T))^2(\log\log(T))^2\big) \\
=& O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\sqrt{\log T}\log(|\mathcal{J}|^{1/4}T)}\right).
\end{aligned} \tag{5.25}$$

Total Regret of RNRM-UCB algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of RNRM-UCB algorithm as:

$$\overline{\text{Regret}}(\text{RNRM-UCB}, T) = \overline{\text{Regret}}_{\text{Warm-up}}(\text{RNRM-UCB}, T) + \overline{\text{Regret}}_{\text{Learning}}(\text{RNRM-UCB}, T) \tag{5.26a}$$

$$= O(|\mathcal{X}|\log(|\mathcal{J}|^{1/4}T)) + O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\sqrt{\log T}\log(|\mathcal{J}|^{1/4}T)}\right). \tag{5.26b}$$

Q.E.D.

Lemma 5.5.1. *For any price* $\mathbf{x} \in \mathcal{X}$*,*

$$\sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau \mathbb{E}\big[\mathbf{Rad}_m(p)\big] = O\left(\sqrt{T_{\mathbf{x}} \log(T)}\right). \tag{5.27}$$

Proof of Lemma 5.5.1. Plugging the definition of $\mathbf{Rad}_m(\mathbf{x}),\ \forall m = 1, \ldots, M$, one has:

$$\begin{aligned}
\sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau \mathbb{E}\big[\mathbf{Rad}_m(\mathbf{x})\big] =& O\left(\sqrt{\log(|\mathcal{J}|^{1/4}T)}\right) \sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau \cdot O\left(\sum_{j \in \mathcal{J}_{\text{on}}(\mathbf{x})} \frac{1}{\sqrt{n_{mj}(\mathbf{x})}}\right) && \text{(5.28a)}\\
\leq& O\left(\sqrt{\log(|\mathcal{J}|^{1/4}T)}\right) \sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau \cdot O\left(\sum_{j \in \mathcal{J}_{\text{on}}(\mathbf{x})} \frac{1}{\sqrt{I_m L\tau}}\right) && \text{(5.28b)}\\
=& O\left(\sqrt{\log(|\mathcal{J}|^{1/4}T)}\right) \sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau \cdot O\left(\frac{|\mathcal{J}|}{\sqrt{I_m L\tau}}\right) && \text{(5.28c)}\\
=& O\left(|\mathcal{J}|\sqrt{\log(|\mathcal{J}|^{1/4}T)}\right) \sum_{m \in \mathcal{M}_{\mathbf{x}}} \sqrt{I_m L\tau} && \text{(5.28d)}\\
\leq& O\left(|\mathcal{J}|\sqrt{\log(|\mathcal{J}|^{1/4}T)}\right) \cdot \sqrt{|\mathcal{M}_{\mathbf{x}}| \sum_{m \in \mathcal{M}_{\mathbf{x}}} I_m L\tau} && \text{(5.28e)}\\
=& O\left(|\mathcal{J}|\sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}} \log(|\mathcal{J}|^{1/4}T)}\right), && \text{(5.28f)}
\end{aligned}$$

where and (5.28b) is by $n_{mj}(\mathbf{x}) = O(I_m L\tau),\ \forall j \in \mathcal{J}_{\text{on}}(\mathbf{x})$ and (5.28e) is by Jensen's inequality.

5.5.3 Performance of RNRM-TS

Before conducting the regret analysis for RNRM-TS, we introduce a new performance measure, termed *Bayesian Regret*, which is typically used for the Bayesian setting in online learning algorithms. Bayesian regret is computed as the expected regret of the instance in expectation over all problem instances. Similarly, we define a relaxed Bayesian regret and it is easy to show this relaxed Bayesian regret is a valid upper bound of the Bayesian regret.

Definition 5.5.2. *Denoting all the events before period* t *by* $\mathcal{F}_t,\ \forall t \geq 0$*, we define the relaxed Bayesian Regret of policy* π *by period* T *as*

$$\overline{BR}(\pi, T) = \mathbb{E}_{\{\alpha_j(\mathbf{x})\}_{j \in \mathcal{J}} \sim \textit{priori}}\big[\mathbb{E}[J^{LP} - J^{\pi} | \mathcal{F}_T]\big].$$

The relaxed Bayesian regret bound of RNRM-TS is in the same order as that we obtain for

RNRM-UCB approach in §5.5.2.

Theorem 5.5.2. *The T-period cumulative Bayesian regret of RNRM-TS is bounded by $O(\sqrt{PT\log(T)})$.*

Proof of Theorem 5.5.2. The analysis here is similar to that in the proof of Theorem 5.5.1. We first bound the Bayesian regret of the Warm-up Phase and then for the Learning Phase.

Bayesian regret in the Warm-up Phase. Similarly, we can bound the Bayesian regret in the Warm-up Phase as

$$\overline{\text{BR}}_{\text{Warm-up}}(\text{RNRM-TS}, T) = O(|\mathcal{X}|\log(|\mathcal{J}|^{1/4}T)). \tag{5.29}$$

Bayesian regret in the Learning Phase. In batch m, the chosen price $\mathbf{x}_m$ and the state-independent optimal price $\tilde{\mathbf{x}}$ are identically distributed conditional on $\mathcal{F}_m$, which indicates that $\mathbb{E}\big[U_m(\mathbf{x}_m) - U_m(\tilde{\mathbf{x}})|\mathcal{F}_m\big] = 0$ (e.g., see the proof of Lemma 3.7 in Slivkins et al. (2019)). Thus, the Bayesian regret in batch m is:

$$\begin{aligned}
\overline{\text{BR}}_m &= \mathop{\mathbb{E}}_{\{\alpha_j(\mathbf{x})\}_{j\in\mathcal{J}}\sim\text{priori}}\Big[\mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}})I_mL\tau - J_m^{\text{RNRM-TS}}(\mathbf{x}_m)|\mathcal{F}_m\big]\Big] && (5.30a)\\
&= \mathop{\mathbb{E}}_{\{\alpha_j(\mathbf{x})\}_{j\in\mathcal{J}}\sim\text{priori}}\Big[\mathbb{E}\big[I_mL\tau U_m(\mathbf{x}_m) - J_m^{\text{RNRM-TS}}(\mathbf{x}_m) + r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}})I_mL\tau - I_mL\tau U_m(\tilde{\mathbf{x}})|\mathcal{F}_m\big]\Big] && (5.30b)\\
&= \mathop{\mathbb{E}}_{\{\alpha_j(\mathbf{x})\}_{j\in\mathcal{J}}\sim\text{priori}}\Big[\mathbb{E}\big[I_mL\tau U_m(\mathbf{x}_m) - J_m^{\text{RNRM-TS}}(\mathbf{x}_m)|\mathcal{F}_m\big]\Big] && (5.30c)\\
&\quad + \mathop{\mathbb{E}}_{\{\alpha_j(\mathbf{x})\}_{j\in\mathcal{J}}\sim\text{priori}}\Big[\mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}})I_mL\tau - I_mL\tau U_m(\tilde{\mathbf{x}})|\mathcal{F}_m\big]\Big] && (5.30d)\\
&= \mathbb{E}\big[I_mL\tau U_m(\mathbf{x}_m) - J_m^{\text{RNRM-TS}}(\mathbf{x}_m)\big] + \mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}})I_mL\tau - I_mL\tau U_m(\tilde{\mathbf{x}})\big] && (5.30e)\\
&\le \mathbb{E}\big[I_mL\tau U_m(\mathbf{x}_m) - I_mL\tau\cdot r(\alpha_{\mathbf{x}_m},\mathbf{x}_m) + O(\tau)\big] + \mathbb{E}\big[I_mL\tau\cdot r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}}) - I_mL\tau U_m(\tilde{\mathbf{x}})\big] && (5.30f)\\
&= I_mL\tau\Big(\mathbb{E}\big[U_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m},\mathbf{x}_m)\big] + \mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big]\Big) + O(\tau). && (5.30g)
\end{aligned}$$

Derived from Proposition 5.4.1, for some $\gamma > 0$, two properties hold (see, e.g., Slivkins et al. 2019):

$$\mathbb{E}[(r(\alpha_{\mathbf{x}},\mathbf{x}) - U_m(\mathbf{x}))^+] \le \frac{\gamma}{|\mathcal{X}|T}, \tag{5.31}$$

$$\mathbb{E}[(L_m(\mathbf{x}) - r(\alpha_{\mathbf{x}},\mathbf{x}))^+] \le \frac{\gamma}{|\mathcal{X}|T}. \tag{5.32}$$

Following these two properties of the stationary revenue rate in (5.31) and (5.32), we have

$$\mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big] \le \mathbb{E}\left[(r(\alpha_{\tilde{\mathbf{x}}},\tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}}))^+\right] \le \sum_{\mathbf{x}\in\mathcal{X}}\mathbb{E}\left[(r(\alpha_{\mathbf{x}},\mathbf{x}) - U_m(\mathbf{x}))^+\right] \le \frac{\gamma}{T}, \tag{5.33}$$

and similarly,

$$\begin{aligned}\mathbb{E}\big[U_m(\mathbf{x}_m) - r(\lambda_{\mathbf{x}_m}, \mu_{\mathbf{x}_m}, \mathbf{x}_m)\big] =& \mathbb{E}\big[2 \cdot \mathbf{Rad}_m(\mathbf{x}_m) + L_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m)\big] \\ \leq& \mathbb{E}\big[2 \cdot \mathbf{Rad}_m(\mathbf{x}_m)\big] + \mathbb{E}\big[\,(L_m(\mathbf{x}) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m))^+\,\big] \\ \leq& \mathbb{E}\big[2 \cdot \mathbf{Rad}_m(\mathbf{x}_m)\big] + \sum_{\mathbf{x}\in\mathcal{X}} \mathbb{E}\Big[\,(L_m(\mathbf{x}) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m))^+\,\Big] \\ \leq& \mathbb{E}\big[2 \cdot \mathbf{Rad}_m(\mathbf{x}_m)\big] + \frac{\gamma}{T}.\end{aligned} \tag{5.34}$$

Summing over all batches $m, \forall m = 1, \ldots, M$, we have the cumulative relaxed Bayesian regret by the end of the Learning Phase as:

$$\overline{\text{BR}}_{\text{Learning}}(\pi_{\text{RNRM-TS}}, T) = \sum_{m=1}^{M} \overline{\text{BR}}_m \tag{5.35a}$$

$$= \sum_{m=1}^{M} \Big(I_m L\tau \Big(\mathbb{E}\big[U_m(\mathbf{x}_m) - r(\alpha_{\mathbf{x}_m}, \mathbf{x}_m)\big] + \mathbb{E}\big[r(\alpha_{\tilde{\mathbf{x}}}, \tilde{\mathbf{x}}) - U_m(\tilde{\mathbf{x}})\big] \Big) + O(\tau) \Big) \tag{5.35b}$$

$$\leq \sum_{m=1}^{M} \Big(I_m L\tau \Big(2\frac{\gamma}{T} + 2\mathbb{E}\big[\mathbf{Rad}_m(\mathbf{x}_m)\big] \Big) \Big) + O\big(\tau \log(T)\big) \tag{5.35c}$$

$$\leq 2\sum_{m=1}^{M} I_m L\tau \frac{\gamma}{T} + 2\sum_{\mathbf{x}\in\mathcal{X}} \sum_{m\in\mathcal{M}_{\mathbf{x}}} I_m L\tau \mathbb{E}\big[\mathbf{Rad}_m(\mathbf{x})\big] + O\big(\tau \log(T)\big) \tag{5.35d}$$

$$= 2\gamma + \leq \sum_{\mathbf{x}\in\mathcal{X}} O\left(|\mathcal{J}| \sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}} \log(|\mathcal{J}|^{1/4} T)} \right) + O\big(\tau \log(T)\big) \tag{5.35e}$$

$$\leq\leq \sum_{\mathbf{x}\in\mathcal{X}} O\left(|\mathcal{J}| \sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}} \log(|\mathcal{J}|^{1/4} T)} \right) + O\big((\log(T))^2 (\log\log(T))^2\big) \tag{5.35f}$$

$$=\leq \sum_{\mathbf{x}\in\mathcal{X}} O\left(|\mathcal{J}| \sqrt{|\mathcal{M}_{\mathbf{x}}| T_{\mathbf{x}} \log(|\mathcal{J}|^{1/4} T)} \right), \tag{5.35g}$$

where (5.35e) is by Lemma 5.5.1 and (5.35f) is by Jensen's inequality.

Regret of RNRM-TS algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of RNRM-TS algorithm as:

$$\overline{\text{BR}}(\text{RNRM-TS}, T) = \overline{\text{BR}}_{\text{Warm-up}}(\text{RNRM-TS}, T) + \overline{\text{BR}}_{\text{Learning}}(\text{RNRM-TS}, T) \tag{5.36a}$$

$$= O(|\mathcal{X}| \log(|\mathcal{J}|^{1/4} T)) + O\left(\sqrt{|\mathcal{J}||\mathcal{X}| T \sqrt{\log T} \log(|\mathcal{J}|^{1/4} T)} \right). \tag{5.36b}$$

Q.E.D.

5.6 Concluding Remarks

We consider a price-based network revenue management problem with multiple products and multiple reusable resources over a finite time horizon. Under a specific offered price vector, upon completion of existing customers under the previously posted price, the future service process of the system can be reduced to a Jackson queueing network consisting with M/M/c queues. Both the mean demand rate of each product and the average usage time for each resource depend on the posted price vector but the mappings are unknown to the firm. We propose two new multi-armed bandit (MAB) based learning and pricing algorithms to maximize the expected total revenue. Our proposed algorithms contain a Warm-up Phase to eliminate the heavy-tail effects, and a Learning Phase to identify the optimal price from a predefined candidate price set. Our algorithms further separate the Learning Phase into successive operational batches and adaptively select a price using past sales for the next batch. We prove that both the proposed algorithms can achieve a cumulative regret $O\left(\sqrt{|\mathcal{J}||\mathcal{X}|T\log T\log(|\mathcal{J}|^{1/4}T)}\right)$, where $|\mathcal{J}|$ is the number of products, $|\mathcal{X}|$ is the cardinality of the feasible price set, and T is the total number of time periods. This regret result matches the lower bound up to a logarithmic factor. As an intermediate step of regret analysis, we also the first to develop a coupling analysis for analyzing the time for a Jackson network to reach the steady state.

There are several plausible future research avenues. First, one may consider generalizing the current model to accommodate general arrival and service distributions. However, this would require different methods in analyzing the regret, especially the loss of nonstationarity, which we shall leave it for future research. Second, this paper considers stationary deman. There are also many literature considering nonstationary demand (see, Borgs et al. 2014, Besbes et al. 2015, Lei and Jasin 2020). Therefore, extending the current model can benefit more real-world applications.

CHAPTER 6

Conclusion

In this dissertation work, we focus on embracing adaptive optimization and learning to develop new decision models and solution approaches for solving decision-making problems under uncertainties in modern service systems, including intelligent transportation and rental services with reusable resources. Specifically, we apply machine learning techniques, including support vector machines (SVM), deep neural networks (DNN), and multi-armed bandits (MAB), to solve problems that arise in the aforementioned modern service systems.

In Chapter 2, we develop a learning-enhanced Benders decomposition algorithm to accelerate the solving process of two-stage stochastic programs, which is widely used to solve operational problems in several service systems. Our results show that SVM cut classifier works effectively for identifying valuable cuts, and the LearnBD algorithm reduces the total solving time of all instances for different problems with various sizes and complexities. In Chapter 3, we consider real-time origin-destination transportation requests and propose a weight learning and context-aware route recommendation system. In particular, we evaluate candidate routes based on WSPP, where the uncertain weights are learned from the historical choices of drivers through deep learning. We evaluate our method and two benchmarks with 4 million requests received by Didi Chuxing in Beijing. Based on the results, we demonstrate that distinguishing request scenarios helps provide preferable context-aware route recommendations.

In Chapter 4 and Chapter 5, we formulate a revenue management problem with reusable resources under a single-product single-resource setting and a multi-product multi-resource setting, respectively. We formulate the revenue management problem as multi-armed bandits (MAB) problems and develop pricing algorithms by leveraging upper confidence bound (UCB) and Thompson sampling (TS) algorithms. We derive theoretical regret bounds with high probability guarantee, which match the lower bound up to a logarithmic factor. As an intermediate step of regret analysis, we are the first to develop a coupling analysis for analyzing the time for a (network) queueing system to reach the steady-state from an empty state or from a steady-state under another set of system parameters. Our experiments demonstrate and confirm the promising numerical performance results of the proposed algorithms.

For future research, we anticipate applying the structure developed in this dissertation, for applying machine learning techniques to enhance the models and solution approaches, to other service system operational problems, especially the adaptive decision-making processes under information uncertainty.

APPENDIX A

Appendix For Chapter 2

A.1 Preliminaries of SVM

We start with

$$f_{SVM}(\mathbf{o}') = \text{sign}\left[\mathbf{w}^T\phi(\mathbf{o}') + b\right] \tag{A.1}$$

and $u = \mathbf{w}^T\phi(\mathbf{o}') + b$, where $\phi(\cdot)$ is a unique mapping function such that $K(\mathbf{o}_1, \mathbf{o}_2) = \langle\phi(\mathbf{o}_1), \phi(\mathbf{o}_2)\rangle$. By the kernel trick (see, e.g., Schölkopf et al. 2002), we do not have to know the exact form of $\phi(\cdot)$ and we can employ the SVM model only with kernel function $K(\cdot,\cdot)$. In Proposition A.1.4, we show this formulation is equivalent to (2.16). With a penalty hyperparameter $C \geq 0$ assigned to the prediction error, the objective function for solving the parameters is defined as $\frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{d=1}^{\Gamma}\xi_d$, where the first term representing the flatness and $\xi_d,\ d = 1, \ldots, \Gamma$ is an auxiliary variable for representing loss amount of training data $(\mathbf{o}_d, l_d)$. The parameters can be solved by an optimization problem:

$$\text{(SVM-P)} \quad \min_{\mathbf{w},\xi,b} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{d=1}^{\Gamma}\xi_d \tag{A.2a}$$

$$\text{s.t.} \ l_d \cdot \left(\mathbf{w}^T\phi(\mathbf{o}_d) + b\right) \geq 1 - \xi_d \qquad d = 1, \ldots, \Gamma; \tag{A.2b}$$

$$\xi_d \geq 0 \qquad d = 1, \ldots, \Gamma, \tag{A.2c}$$

where (A.2b) are used to calculate the hinge loss and (A.2c) are sign restrictions of ξ. (SVM-P) is a convex optimization problem with convex inequality constraints and a quadratic objective function, and thus it is easy to solve by taking Lagrangian Dual and applying Krash-Kuhn-Tucker (KKT) conditions (see, e.g., Chang and Lin 2011).

Proposition A.1.1. *The optimal objective value of*

$$\min_{\mathbf{w},\xi,b} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{d=1}^{\Gamma}\xi_d - \sum_{d=1}^{\Gamma} a_d\left\{l_d \cdot \left(\mathbf{w}^T\phi(\mathbf{o}_d) + b\right) - 1 + \xi_d\right\} - \sum_{d=1}^{\Gamma} v_d\xi_d \tag{A.3}$$

with any $\mathbf{a}, \mathbf{v} \geq 0$ *is a valid lower bound of (SVM-P).*

Proof. Assume that $(\mathbf{w}_1, \xi_1, b_1)$ is an optimal solution to (SVM-P), and therefore it is feasible to the relaxation (A.3). Given the constraints in (SVM-P), we have $l_d \cdot \left(\mathbf{w}_1^T \phi(\mathbf{o}_d) + b_1\right) - 1 + \xi_{1d} \geq 0$ and $\xi_{1d} \geq 0$ for all d. Therefore, for any $\mathbf{a}, \mathbf{v} \geq 0$, the objective value of (A.3) based on solution $(\mathbf{w}_1, \xi_1, b_1)$ is no larger than $\frac{1}{2}\mathbf{w}_1^T \mathbf{w}_1 + C \sum_{d=1}^{\Gamma} \xi_{1d}$. Moreover, as the optimal objective value of (A.3) is smaller than or equal to the objective value of any feasible solution, we can conclude that the objective value of (A.3) evaluated at the feasible solution $(\mathbf{w}_1, \xi_1, b_1)$ is always smaller than or equal to $\frac{1}{2}\mathbf{w}_1^T \mathbf{w}_1 + C \sum_{d=1}^{\Gamma} \xi_{1d}$, which is the optimal objective value of (SVM-P) and thus provides a valid lower bound of (SVM-P). This completes the proof. □

By associating dual variables $\mathbf{a} \geq 0$ with inequality constraints (A.2b) and dual variables $\mathbf{v} \geq 0$ with inequality constraints (A.2c), we can relax those two sets of constraints and then obtain the corresponding Lagrangian function for any feasible solution $(\mathbf{w}, \xi, b)$ as

$$L(\mathbf{w}, \xi, b; \mathbf{a}, \mathbf{v}) = \frac{1}{2}\mathbf{w}^T \mathbf{w} + C \sum_{d=1}^{\Gamma} \xi_d - \sum_{d=1}^{\Gamma} a_d \left\{ l_d \cdot \left(\mathbf{w}^T \phi(\mathbf{o}_d) + b\right) - 1 + \xi_d \right\} - \sum_{d=1}^{\Gamma} v_d \xi_d.$$

By weak duality, the Lagrangian problem

$$\min_{\mathbf{w}, \xi, b} L(\mathbf{w}, \xi, b; \mathbf{a}, \mathbf{v})$$

yields a valid lower bound of (SVM-P). Moreover,

$$\max_{\mathbf{a} \geq \mathbf{0}, \mathbf{v} \geq \mathbf{0}} \min_{\mathbf{w}, \xi, b} L(\mathbf{w}, \xi, b; \mathbf{a}, \mathbf{v})$$

is the dual problem that seeks the best lower bound.

Definition A.1.1. *Krash-Kuhn-Tucker (KKT) is a set of conditions including: Primal feasibility, dual feasibility, complementary slackness, and the first derivative of Lagrangian function* $L(\cdot)$ *being zero. If the primal problem is*

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \textit{subject to} \quad & f_i(x) \leq 0 \ \ \forall i \in I \\ & h_i(x) = 0 \ \ \forall i \in I' \end{aligned}$$

and the associated dual multipliers are $\lambda \geq 0$ *and* μ*, then the KKT conditions are:*

- $f_i(x^*) \leq 0 \ \ \forall i \in I$ *and* $h_i(x^*) = 0 \ \ \forall i \in I'$ *(primal feasibility),*
- $\lambda^* \geq 0$ *(dual feasibility),*

- $f_i(x^*)\lambda_i^* = 0$ *(complementary slackness),*
- $\nabla f_0(x^*) + \sum_{i\in I} \lambda_i^* \nabla f_i(x^*) + \sum_{i\in I'} \mu_i^* \nabla h_i(x^*) = 0$ *(first derivative of $L(\cdot)$ is zero).*

Proposition A.1.2. *The strong duality holds for (SVM-P) and KKT conditions are satisfied at the optimal primal and dual solution pair.*

Proof. (SVM-P) has a quadratic objective and affine inequality constraints, and therefore by Slater's condition strong duality holds. Because (SVM-P) is differentiable, KKT conditions hold at the global optimum. This completes the proof. □

Theorem A.1.1. *The optimal objective value of the optimization problem*

$$\max_{\mathbf{a}\geq\mathbf{0},\mathbf{v}\geq\mathbf{0}} \min_{\mathbf{w},\xi,b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{d=1}^{\Gamma}\xi_d - \sum_{d=1}^{\Gamma} a_d \left\{ l_d \cdot \left(\mathbf{w}^T\phi(\mathbf{o}_d) + b\right) - 1 + \xi_d \right\} - \sum_{d=1}^{\Gamma} v_d\xi_d \tag{A.4}$$

equals to the optimal objective value of (SVM-P).

Proof. Recall the Lagrangian dual problem

$$\max_{\mathbf{a}\geq\mathbf{0},\mathbf{v}\geq\mathbf{0}} \min_{\mathbf{w},\xi,b} L(\mathbf{w},\xi,b;\mathbf{a},\mathbf{v}).$$

By Proposition A.1.2, strong duality holds and thus the optimal objective value of the dual problem and primal problem are equal. □

Proposition A.1.3. *The Lagrangian dual function* (A.3) *in Proposition A.1.1 can be reformulated as*

$$\frac{1}{2}\sum_{d=1}^{\Gamma}\sum_{d'=1}^{\Gamma} l_d l_{d'} a_d a_{d'} K(\mathbf{o}_d, \mathbf{o}_{d'}) + \sum_{d=1}^{\Gamma} a_d - \sum_{d=1}^{\Gamma} v_d\xi_d \tag{A.5a}$$

$$\textit{with} \quad \sum_{d=1}^{\Gamma} a_d l_d = 0; \tag{A.5b}$$

$$C - a_d - v_d = 0 \quad d = 1,\ldots,\Gamma. \tag{A.5c}$$

Proof. The Lagrangian dual function (A.3) is differentiable, and therefore the derivatives as-

sociated with $(\mathbf{w}, \xi, b)$ at the minimum are equal to zero, i.e.,

$$\frac{\partial L}{\mathbf{w}} = 0 \ \rightarrow \ \mathbf{w} = \sum_{d=1}^{\Gamma} a_d l_d \phi_d \tag{A.6a}$$

$$\frac{\partial L}{b} = 0 \ \rightarrow \ \sum_{d=1}^{\Gamma} a_d l_d = 0 \tag{A.6b}$$

$$\frac{\partial L}{\xi} = 0 \ \rightarrow \ C - a_d - v_d = 0, \ d = 1, \ldots, \Gamma \ \rightarrow \ a_d \leq c, \ d = 1, \ldots, \Gamma. \tag{A.6c}$$

Plugging in the results in (A.6), we can obtain the reformulation of (A.3) in (A.5). This completes our proof. □

Theorem A.1.2. *The Lagrangian dual problem* (A.4) *is equivalent to solving a convex quadratic program:*

$$\max_{\mathbf{a}} \ \frac{1}{2} \sum_{d=1}^{\Gamma} \sum_{d'=1}^{\Gamma} l_d l_{d'} a_d a_{d'} K(\mathbf{o}_d, \mathbf{o}_{d'}) + \sum_{d=1}^{\Gamma} a_d \tag{A.7a}$$

$$s.t. \ \sum_{d=1}^{\Gamma} a_d l_d = 0; \tag{A.7b}$$

$$0 \leq a_d \leq C \quad d = 1, \ldots, \Gamma. \tag{A.7c}$$

Proof. By Proposition A.1.3, we obtain an equivalent formulation of (A.4) as follows.

$$\max_{\mathbf{a}, \mathbf{v} \geq 0} \ \frac{1}{2} \sum_{d=1}^{\Gamma} \sum_{d'=1}^{\Gamma} l_d l_{d'} a_d a_{d'} K(\mathbf{o}_d, \mathbf{o}_{d'}) + \sum_{d=1}^{\Gamma} a_d - \sum_{d=1}^{\Gamma} v_d \xi_d \tag{A.8a}$$

$$\text{with} \ \sum_{d=1}^{\Gamma} a_d l_d = 0; \tag{A.8b}$$

$$C - a_d - v_d = 0 \quad d = 1, \ldots, \Gamma; \tag{A.8c}$$

In the third term in the objective function (A.8a), all $v_d \xi_d, \ \forall d = 1, \ldots, \Gamma$ are zero at the optimum because of the complementary slackness by Proposition A.1.2. Therefore, we can discard the third term without loss of optimality. Moreover, because $v_d \geq 0, \ \forall d = 1, \ldots, \Gamma$, we can combine (A.8c) with $\mathbf{v} \geq 0$ and derive valid constraints $a_d \leq C, \ \forall d = 1, \ldots, \Gamma$, which helps to eliminate variables $v_d, \ \forall d = 1, \ldots, \Gamma$. Finally, we can rewrite model (A.8) as shown in (A.7) (see, e.g., Chang and Lin 2011). This completes our proof. □

Proposition A.1.4. *The parameter of the classifier in* (A.1) *are* $\mathbf{w}^* = \sum_{d=1}^{\Gamma} a_d^* l_d \phi_d$ *and* $b^* = 1 - \sum_{d'=1}^{\Gamma} l_{d'} (a_{d'}^* K(\mathbf{o}_d, \mathbf{o}_{d'}))$ *for any* $d = 1, \ldots, \Gamma$ *associated with* $a_d^* \in (0, C)$. *The three prediction*

functions (2.14), (2.16) *and* (A.1) *are equivalent to each other, where the support vector set* S *in* (2.16) *contains all* $(\mathbf{o}_d, l_d),\ d = 1, \ldots, \Gamma$ *such that* $a_d^* > 0$.

Proof. The value of $\mathbf{w}^*$ is obtained by (A.6a) and it shows the equivalence between (2.14) and (A.1). Assume that we solve and obtain an optimal solution $\mathbf{a}^*$ to (A.7). Then following the complementary slackness:

$$a_d \left[l_d \cdot \left(\mathbf{w}^T \phi(\mathbf{o}_d) + b \right) - 1 + \xi_d \right] = 0,\ v_d \xi_d = 0,\ \forall d = 1, \ldots, \Gamma,$$

we have:

- If $a_d^* = C > 0$, then $l_d \cdot \left(\mathbf{w}^{*T} \phi(\mathbf{o}_d) + b \right) = 1 - \xi_d^*$. By $a_d^* = C - v_d^*$ we have $v_d^* = 0$ and thus $\xi_d^* \geq 0$. The observation d is called non-margin support vector.
- If $0 < a_d^* < C$, then $l_d \cdot \left(\mathbf{w}^{*T} \phi(\mathbf{o}_d) + b \right) = 1 - \xi_d^*$. Similarly, we have $v_d^* > 0$ and thus $\xi_d^* = 0$. The observation d is called margin support vector. Therefore, we can compute $b^* = 1 - \sum_{d'=1}^{\Gamma} l_{d'} (a_{d'}^* K(\mathbf{o}_d, \mathbf{o}_{d'}))$ with any $d = 1, \ldots, \Gamma$ associated with $a_d^* \in (0, C)$,
- If $a_d^* = 0$, then this type of observation d does not affect the value of the second prediction function. Therefore, we can build a support vector set S of $(\mathbf{o}_d, l_d),\ d = 1, \ldots, \Gamma$ with $a_d^* \neq 0$ and thus simplify (2.14) as (2.16).

□

A.2 Detailed Formulations of Problems for Computational Studies

A.2.1 CFLP

Consider a set W of production plants (facilities) and a set F of factories which have uncertain demand $\tilde{d}$. The setup cost of facility $i,\ \forall i \in W$ is k_i and the production capacity limit is u_i. The demand of factory $j, \forall j \in F$ is uncertain and can be satisfied by products produced in facility $i,\ \forall i \in W$ if it is open with a unit transportation cost c_{ij}, and the unmet demand will generate lost-sale with a unit penalty cost ρ_j. One needs to decide a subset of facilities to open before the realization of the demand to minimize the expected total cost.

The two-stage stochastic programming model consists of two types of decisions. We define first-stage binary decision variables $x_i,\ \forall i \in W$ such that $x_i = 1$ if we open facility i and $x_i = 0$ otherwise. In the second stage, we obtain the demand value from each factory and define continuous decision variables $y_{ij} \geq 0,\ \forall i \in W,\ j \in F$, which represent transportation units from facility i

to factory j. The model aims to find the best decisions to minimize the facility setup cost, expected transportation cost, and expected lost-sale cost. The first-stage formulation is:

$$\begin{aligned} \text{(CFLP)} \quad \min_{x} \quad & \sum_{i \in W} k_i x_i + \sum_{\omega \in \Omega} p_\omega Q_\omega(x) \\ \text{s.t.} \quad & x_i \in \{0,1\} \quad i \in W. \end{aligned} \tag{A.9}$$

The second-stage problem for each scenario ω is defined using variables $y_{ij},\ i \in W,\ j \in F$ and auxiliary variables $\alpha_j,\ j \in F$ that denote the amount of unmet demand. We have

$$\begin{aligned} Q_\omega(x) \ = \ \min_{y,\alpha} \quad & \sum_{i \in W} \sum_{j \in F} c_{ij} y_{ij} + \sum_{j \in F} \rho_j \alpha_j \\ \text{s.t.} \quad & \sum_{j \in F} y_{ij} \leq u_i x_i \qquad i \in W; \\ & \tilde{d}_{\omega,j} - \sum_{i \in W} y_{ij} \leq \alpha_j \quad j \in F; \\ & y_{ij} \geq 0 \qquad i \in W,\ j \in F; \\ & \alpha_j \geq 0 \qquad j \in F. \end{aligned} \tag{A.10}$$

By allowing unmet demand, the problem always has a feasible solution and Benders decomposition only generates optimality cuts. Then, we derive the dual of second-stage problems and formulate SPs as shown in Section 2.1.1. By defining dual variables $h_i,\ \forall i \in W$ and $\pi_j,\ \forall j \in F$, respectively associated with the first and second constraints in model (A.10), we formulate the subproblem in scenario ω as

$$\begin{aligned} (\text{SP}_\omega) \qquad \max_{h,\pi} \quad & -\sum_{i \in W} u_i x_i h_i + \sum_{j \in F} \tilde{d}_{\omega,j} \pi_j \\ \text{s.t.} \quad & -h_i - \pi_j \leq c_{ij} \ \ i \in W,\ j \in F; \\ & 0 \leq \pi_j \leq \rho \qquad j \in F; \\ & h_i \geq 0 \qquad i \in W. \end{aligned} \tag{A.11}$$

Letting $V^{\omega,t}$ be a collection of extreme points of SP_ω that have been identified when reaching iteration t, we formulate

$$\begin{aligned} (\text{RMP}^t) \quad \min_{x,\theta} \quad & \sum_{i \in W} k_i x_i + \sum_{\omega \in \Omega} p_\omega \theta_\omega \\ \text{s.t.} \quad & \theta_\omega \geq -\sum_{i \in W} u_i x_i \hat{h}_i + \sum_{j \in F} \tilde{d}_{\omega,j} \hat{\pi}_j \quad (\hat{h}_i, \hat{\pi}_j) \in V^{\omega,t},\ \omega \in \Omega; \\ & x_i \in \{0,1\} \qquad i \in W. \end{aligned} \tag{A.12}$$

A.2.2 CMND

Consider a directed network with node set N, arc set A, and commodity set K. An uncertain $\tilde{v}_k$ amount of commodity $k, \forall k \in K$ must be routed from an origin node, $o_k \in N$, to a destination node, $d_k \in N$. The installation cost and arc capacity of arc (i,j), $\forall (i,j) \in A$ are f_{ij} and u_{ij}, respectively. The cost for transporting one unit of commodity k, $\forall k \in K$ on installed arc (i,j), $\forall (i,j) \in A$ is c_{ij}^k. One needs to decide a subset of arcs to install before the realization of the demand to minimize the expected total cost. In the first stage, we make binary decisions x_{ij}, $\forall (i,j) \in A$ such that $x_{ij} = 1$ if we install arc (i,j). In the second stage, we obtain the demand of each commodity and then solve non-negative continuous decisions $y_{ij}^k, \forall (i,j) \in A,\ k \in K$, which represents transportation units of commodity k on arc (i,j).

The first-stage formulation is:

$$\begin{aligned} \text{(CMND)} \quad \min_{x} \quad & \sum_{(i,j)\in A} f_{ij} x_{ij} + \sum_{\omega \in \Omega} p_\omega Q_\omega(x) \\ \text{s.t.} \quad & x_{ij} \in \{0,1\} \quad (i,j) \in A. \end{aligned} \tag{A.13}$$

The second-stage problem for each scenario ω is defined with decision variables y_{ij}^k, $\forall (i,j) \in A,\ k \in K$ and auxiliary variables α_i^k, $\forall i \in N,\ k \in K$ for denoting unmet demand:

$$\begin{aligned} Q_\omega(x) \ = \ \min_{y,\alpha} \quad & \sum_{(i,j)\in A} \left[\sum_{k \in K} c_{ij}^k y_{ij}^k + B\alpha_i^k \right] \\ \text{s.t.} \quad & \sum_{j:(j,i)\in A} y_{ji}^k - \sum_{j:(i,j)\in A} y_{ij}^k \le \tilde{d}_i^k + \alpha_i^k && i \in N,\ k \in K; \\ & \sum_{k \in K} y_{ij}^k \le u_{ij} x_{ij} && (i,j) \in A; \\ & y_{ij}^k \ge 0 && (i,j) \in A,\ k \in K; \\ & \alpha_i^k \ge 0 && i \in N,\ k \in K. \end{aligned} \tag{A.14}$$

Define an auxiliary demand unmet cost B. The parameter $\tilde{d}_i^k$ is set to $\tilde{v}^k$ if node i is the origin of the commodity k, $-\tilde{v}^k$ is node i is the destination of the commodity k, or 0 otherwise.

By allowing unmet demand, the problem always has a feasible solution and Benders decomposition only generates optimality cuts. We derive the dual of second-stage problems and formulate SPs as shown in Section 2.1.1. By defining dual variables h_i^k, $\forall i \in N,\ k \in K$ and π_{ij}, $\forall (i,j) \in A$, respectively associated with the first and second constraints in model (A.14), we formulate the sub-

problem in scenario ω as

$$
\begin{aligned}
(\text{SP}_\omega) \qquad & \max_{h,\pi} \sum_{i\in N, k\in K} -\tilde{d}_i^k h_i^k - \sum_{(i,j)\in A} u_{ij} x_{ij} \pi_{ij} \\
& \text{s.t. } h_i^k - h_j^k - \pi_{ij} \le c_{ij}^k && (i,j)\in A,\ k\in K; \\
& 0 \le \pi_{ij} && (i,j)\in A. \\
& 0 \le h_i^k \le B && i\in N,\ k\in K.
\end{aligned} \tag{A.15}
$$

Letting $V^{\omega,t}$ be a collection of extreme points of SP_ω that have been identified when reaching iteration t, we formulate

$$
\begin{aligned}
(\text{RMP}^t) \quad & \min_{x,\theta} \sum_{(i,j)\in A} f_{ij} x_{ij} + \sum_{\omega\in\Omega} p_\omega \theta_\omega \\
& \text{s.t. } \theta_\omega \ge \sum_{i\in N, k\in K} \tilde{d}_i^k \hat{h}_i^k - \sum_{(i,j)\in A} u_{ij} x_{ij} \hat{\pi}_{ij} && (\hat{h}_i, \hat{\pi}_{ij}) \in V^{\omega,t},\ \omega\in\Omega; \\
& x_{ij} \in \{0,1\} && (i,j)\in A.
\end{aligned} \tag{A.16}
$$

APPENDIX B

Appendix For Chapter 4

B.1 Summary of Major Notation

Table B.1: Summary of Major Notation for Problem Formulation

$\mathcal{P}$	the set of candidate prices, $P = \lvert\mathcal{P}\rvert$
T	the number of periods in consideration
c	the capacity of reusable resources
n	the number of customers in the system
λ_p	the arrival rate under price $p \in \mathcal{P}$
μ_p	the service rate of one unit of resource under price $p \in \mathcal{P}$
k_p	the identical threshold length of the queue for balking under price $p \in \mathcal{P}$
η_p	the probability for customers to join the queue when balking happens under price $p \in \mathcal{P}$
ν_p	the reneging rate when reneging happens under price $p \in \mathcal{P}$
ℓ_{np}	the probability that the system has n customers in the steady state under price $p \in \mathcal{P}$
$\pi(n,t)$	the pricing policy $\pi : \{(n,t) : n = 0, 1, \ldots, \infty, t = 1, \ldots, T\} \to \mathcal{P}$, where the firm selects price $\pi(n,t) \in \mathcal{P}$ when there are n customers in the system at the beginning of period t
J_t^π	the expected revenue under pricing policy π during period t, $\forall t = 1, \ldots, T$,
J^π	the expected revenue over periods $\{1, \ldots, T\}$, $J^\pi = \sum_{t=1}^{T} J_t^\pi$
π^*	the state-dependent optimal policy $\pi^* = \text{argmax}_\pi J^\pi$
J^*	the state-dependent optimal expected revenue $J^* = J^{\pi^*}$
$\tilde{p}$	the optimal state-independent price, $\tilde{p} = \text{argmax}_{p \in \mathcal{P}} \frac{\lambda_p}{\mu_p} p$
J^{LP}	the optimal objective value of LP defined in (4.2), $J^{\text{LP}} = T \frac{\lambda_{\tilde{p}}}{\mu_{\tilde{p}}} \tilde{p}$
$r(\lambda_p, \mu_p, p)$	the stationary revenue rate of price $p \in \mathcal{P}$, $r(\lambda_p, \mu_p, p) = \frac{\lambda_p}{\mu_p} p$

Table B.2: Summary of Major Notation for BUCB and BTS

M	the number of batches
I_m	batch $m,\ m = 1, \ldots, M$ contains $I_m \tau$ periods, where $I_m = 2^m$ and $\tau = \log(T)(\log\log(T))^2$
$n_m(p)$	the number of arrived customers under price p by the end of batch m
$n_m^s(p)$	the number of customers have been successfully serviced under price p by the end of batch m
$\hat{d}_i(p)$	the observed time intervals between arrivals of customer i and customer $i-1$ under price $p \in \mathcal{P}$
$\breve{d}_m(p)$	an empirical estimate of arrival rate by the end of batch m, $\breve{d}_m(p) = \frac{\sum_{i=1}^{n_m(p)} 1/\hat{d}_i(p)}{n_m(p)}$
$\hat{g}_i(p)$	the observed service time of customer i under price $p \in \mathcal{P}$
$\bar{g}_m(p)$	the empirical mean of service time by the end of batch m, $\bar{g}_m(p) = \frac{\sum_{i=1}^{n_m^s(p)} \hat{g}_i(p)}{n_m^s(p)}$
$n_m^c(p)$	the number of arrived customers when the system has more than $k_p + 1$ customers under price $p \in \mathcal{P}$
$n_m^b(p)$	the number of arrived customers who balk under price $p \in \mathcal{P}$
$n_m^r(p)$	the number of customers who join the queue but leave before getting serviced under price $p \in \mathcal{P}$
$\hat{h}_i(p)$	the observed waiting time that a customer get impatient and leave the queue under price $p \in \mathcal{P}$
$\breve{h}_m(p)$	an empirical estimate of reneging rate by the end of batch m, $\breve{h}_m(p) = \frac{\sum_{i=1}^{n_m^r(p)} \frac{1}{\hat{h}_i(p)}}{n_m^r(p)}$
$\hat{\ell}_p^{br}$	the portion of the time when the number of customers in the system is greater than or equal to $k_p + 1$ after the first 2τ periods in each batch
p_m	the offered price in batch m by the algorithm
$J_m(p)$	the practical revenue collected in batch m with price p
$U_m(p)$	the upper confidence bound of expected stationary revenue rate of price p by the end of batch m
$L_m(p)$	the lower confidence bound of expected stationary revenue rate of price p by the end of batch m
$\mathcal{M}_p \subseteq \{1, \ldots, M\}$	the batches choosing price p
$T_p = \sum_{m \in \mathcal{M}_p} I_m \tau$	the number of periods choosing price p
Gamma$(\alpha^{\lambda_p}, \beta^{\lambda_p})$	the prior/posterior distribution of λ_p
Gamma$(\alpha^{\mu_p}, \beta^{\mu_p})$	the prior/posterior distribution of μ_p
Gamma$(\alpha^{\nu_p}, \beta^{\nu_p})$	the prior/posterior distribution of ν_p
Beta$(\alpha^{\eta_p}, \beta^{\eta_p})$	the prior/posterior distribution of η_p
$\mathcal{C}_{2\tau}$	the event that the system reaches the steady state under the new price within the first 2τ periods
$\bar{\mathcal{C}}_{2\tau}$	the complementary event of $\mathcal{C}_{2\tau}$ that the system does not reach the steady state within 2τ periods

Table B.3: Summary of Major Notation for Coupling Analysis

S_t	the number of customers in the target system at time t, starting from the empty state or from the steady state under another price
S_∞	the number of customers in the system in steady state
$\hat{S}_t$	an auxiliary stochastic process starting with the steady-state distribution and remains thereafter, $\hat{S}_t = S_\infty,\ t \geq 0$
A_n	the busy period between n and $n-1$ customers, $A_n = \min\{t$: n customers in the system at time 0^+ and $n-1$ customers in the system at time $t\}$, $\forall n \geq 1$. $\bar{A}_1$ and σ_1 are the mean and the standard deviation of A_1.
τ	the coupling time, $\tau = \log(T)(\log\log(T))^2$
$\mathcal{A}_t$	the event that system S_t and system $\hat{S}_t$ couple after time t, $\mathcal{A}_t = \{S_t = \hat{S}_t$ after $t\}$
$\mathcal{B}_n^a$	the event where a new customer arrives before any current customers leave starting from n customers in the system
$\mathcal{B}_n^l$	the event where a current customer leaves before any new customer arrive, starting from n customers in the system
$\mathcal{C}_t$	the event where S reaches the steady state under the new price before $t,\ t \in \{\tau, 2\tau\}$
$\bar{\mathcal{C}}_t$	the complementary events of $\mathcal{C}_t,\ t \in \{\tau, 2\tau\}$
$\breve{S}$	the virtual system which has a slower or equal coupling speed as the target system S
$\breve{\mathcal{C}}_t$	the event where the virtual system $\breve{S}$ reaches the steady state under the new price before $t,\ t \in \{\tau, 2\tau\}$

Table B.4: Summary of Major Notation for BLinUCB

$[p_L, p_U]$	a fixed price range that the firm can choose price from
d_f	the dimension of the features
$\mathbf{x}_p$	feature vector of price $p \in [p_L, p_U]$, $\mathbf{x}_p \in \mathbb{R}^{d_f}$
θ_λ	unknown coefficient of the linear relationship between the price feature factor and the average arrival time of the arrival Poisson process, $\theta_\lambda \in \mathbb{R}^{d_f}$ and $\frac{1}{\lambda_p} = \theta_\lambda^T \mathbf{x}_p$
θ_μ	unknown coefficient of the linear relationship between the price feature factor and the average service time of the service Poisson process, $\theta_\mu \in \mathbb{R}^{d_f}$ and $\frac{1}{\mu_p} = \theta_\mu^T \mathbf{x}_p$
ϵ_p	random error term in linear relationship $\bar{d}_p = \theta_\lambda^T \mathbf{x}_p + \epsilon_p$, which follows SE $\left(4/(n_m(p)\lambda_p^2), 2/(n_m(p)\lambda_p)\right)$ (we refer $\bar{d}_p$ and $n_m(p)$ to Table B.1)
N	the number of implemented prices
Ω	a diagonal matrix where the ith element is the variance of the error term of the ith implemented price, i.e., $1/(n_m(p_i)\lambda_{p_i}^2)$.
$\hat{\Omega}_\lambda$	an estimation of Ω where the ith element is computed by $\bar{d}_{p_i}^2/(n_m(p_i))$
$\hat{\Omega}_\mu$	an estimation of Ω_μ where the ith element is computed by $\bar{g}_{p_i}^2/(n_m^s(p_i))$
$\mathbf{X}$	a matrix denoting the features of implemented prices, $\mathbf{X} \in \mathbb{R}^{N \times d_f}$
$\mathbf{d}$	a vector denoting the empirical arrival time means of implemented prices, $\mathbf{d} \in \mathbb{R}^N$
$\mathbf{y}_\mu$	a vector denoting the empirical service time means of implemented prices, $\mathbf{y}_\mu \in \mathbb{R}^N$
$\hat{\theta}_\lambda$	estimated coefficient for the arrival process, $\hat{\theta}_\lambda = (\mathbf{X}^T \hat{\Omega}_\lambda^{-1} \mathbf{X})^{-1} \mathbf{X}^T \hat{\Omega}_\lambda^{-1} \mathbf{d}$
$\hat{\theta}_\mu$	estimated coefficient for the service process, $\hat{\theta}_\mu = (\mathbf{X}^T \hat{\Omega}_\mu^{-1} \mathbf{X})^{-1} \mathbf{X}^T \hat{\Omega}_\mu^{-1} \mathbf{y}_\mu$
$\|\mathbf{x}\|_M$	matrix norm of vector $\mathbf{x}$ induce by a positive semidefinite matrix M, $\|\mathbf{x}\|_M = \sqrt{x^T M x}$
$\mathcal{M}_{\lambda,m}$	a information matrix for the arrival process that is obtained at the beginning of batch m, $\mathcal{M}_{\lambda,m} = \mathbf{X}_m^T \hat{\Omega}_{\lambda,m}^{-1} \mathbf{X}_m$
$\mathcal{M}_{\mu,m}$	a information matrix for the service process that is obtained at the beginning of batch m, $\mathcal{M}_{\mu,m} = \mathbf{X}_m^T \hat{\Omega}_{\mu,m}^{-1} \mathbf{X}_m$
$U_c(p)$	upper confidence bound of the revenue rate of price p, $U_c(p) = \left(\frac{\hat{\theta}_{\mu,m}^T \mathbf{x}}{\hat{\theta}_{\lambda,m}^T \mathbf{x}} + \frac{\sqrt{32\log(T)}}{\hat{\theta}_{\lambda,m}^T \mathbf{x}} \left(r_{\max} \|\mathbf{x}\|_{\mathcal{M}_{\lambda,m}^{-1}} + \|\mathbf{x}\|_{\mathcal{M}_{\mu,m}^{-1}}\right)\right) p$
$\mathcal{C}_\lambda$	a constant such that $\lambda_p \|\mathbf{x}_p\|_2 \leq \mathcal{C}_\lambda$ for all price $p \in [p_L, p_U]$
$\mathcal{C}_\mu$	a constant such that $\mu_p \|\mathbf{x}_p\|_2 \leq \mathcal{C}_\mu$ for all price $p \in [p_L, p_U]$
l_λ	a constant such that the smallest eigenvalue of $\sum_{p \in \mathcal{P}_b} \lambda_m^2 \mathbf{x}_p \mathbf{x}_p^T$ is lower-bounded by l_λ
l_μ	a constant such that the smallest eigenvalue of $\sum_{p \in \mathcal{P}_b} \mu_m^2 \mathbf{x}_p \mathbf{x}_p^T$ is lower-bounded by l_μ

B.2 Proofs in §4.3.2

Proof of Proposition 4.3.1. The proof is based on the concentration inequality for sub-exponential random variables. We show the intermediate results in Lemmas B.2.1 to B.2.3. By definitions of $n_m(p)$ and $n_m^s(p)$, one has $n_m(p) \geq n_m^s(p)$ and thus

$$
\begin{aligned}
&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \\
=&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4r_{\max}\sqrt{\frac{2\log(T)}{n_m^s(p)}} + 4\bar{g}_m(p)\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \\
\geq&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4r_{\max}\sqrt{\frac{2\log(T)}{n_m^s(p)}} + 4\bar{g}_m(p)\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right) \\
=&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \lambda_p\bar{g}_m(p) + \lambda_p\bar{g}_m(p) - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4r_{\max}\sqrt{\frac{2\log(T)}{n_m^s(p)}} + 4\bar{g}_m(p)\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right) \\
=&\mathbb{P}\left(\lambda_p\left|\frac{1}{\mu_p} - \bar{g}_m(p)\right| + \bar{g}_m(p)\left|\lambda_p - \breve{d}_m(p)\right| \leq 4r_{\max}\sqrt{\frac{2\log(T)}{n_m^s(p)}} + 4\bar{g}_m(p)\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right) \\
\geq&\mathbb{P}\left(\lambda_p\left|\frac{1}{\mu_p} - \bar{g}_m(p)\right| \leq r_{\max}\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)\cdot\mathbb{P}\left(\bar{g}_m(p)\left|\lambda_p - \breve{d}_m(p)\right| \leq 4\bar{g}_m(p)\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right) \\
=&\mathbb{P}\left(\left|\frac{1}{\mu_p} - \bar{g}_m(p)\right| \leq \frac{4r_{\max}}{\lambda_p}\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)\cdot\mathbb{P}\left(\left|\lambda_p - \breve{d}_m(p)\right| \leq 4\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right).
\end{aligned}
\tag{B.1}
$$

Given $n_m(p) \geq n_m^s(p) \geq 8\log(T)$ and $r_{\max} \geq \max_{p\in\mathcal{P}} \lambda_p/\mu_p$, by Lemma B.2.2 and Lemma B.2.3, one has

$$
\mathbb{P}\left(\left|\frac{1}{\mu_p} - \bar{g}_m(p)\right| \leq \frac{4r_{\max}}{\lambda_p}\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \geq \mathbb{P}\left(\left|\frac{1}{\mu_p} - \bar{g}_m(p)\right| \leq \frac{4}{\mu_p}\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \geq 1 - \frac{2}{T^4},
$$

and

$$
\mathbb{P}\left(\left|\lambda_p - \breve{d}_m(p)\right| \leq 4\breve{d}_m(p)\sqrt{\frac{2\log(T)}{n_m(p)}}\right) \geq 1 - \frac{2}{T^4}.
$$

Therefore, one can derive the result that:

$$
\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p} - \bar{g}_m(p)\breve{d}_m(p)\right| \leq 4\left(r_{\max} + \bar{g}_m(p)\breve{d}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \geq \left(1 - \frac{2}{T^4}\right)^2 \geq 1 - \frac{4}{T^4}.
$$

Q.E.D.

Lemma B.2.1. ***(Sub-exponential Property.)*** *If a random variable X follows an exponential distribution with mean $1/\lambda$, then the random variable $X - \frac{1}{\lambda}$ is $(\frac{4}{\lambda^2}, \frac{2}{\lambda})$-sub-exponential.*

Proof of Lemma B.2.1. According to the definition of sub-exponential random variables (e.g., in Boucheron et al. 2013), we need to prove that

$$\mathbb{E}\left[\exp\left(s\left(X - \frac{1}{\lambda}\right)\right)\right] \leq \exp\left(\frac{s^2 \frac{4}{\lambda^2}}{2}\right), \ \ \forall s \leq \frac{\lambda}{2}.$$

By the moment generating function of exponential distribution, we have for $s \leq \frac{\lambda}{2} < \lambda$:

$$\mathbb{E}\left[\exp\left(s\left(X - \frac{1}{\lambda}\right)\right)\right] = \mathbb{E}\left[\exp\left(sX\right)\right] \cdot \exp\left(-\frac{s}{\lambda}\right) = \frac{\lambda}{\lambda - s} \cdot \exp\left(-\frac{s}{\lambda}\right).$$

Let $t = \frac{s}{\lambda}$, we have:

$$\log\left(\frac{\lambda}{\lambda - s} \cdot \exp\left(-\frac{s}{\lambda}\right)\right) = -\log(1-t) - t \leq 2t^2 = \log\left(\exp\left(\frac{s^2 \frac{4}{\lambda^2}}{2}\right)\right).$$

Q.E.D.

Lemma B.2.2. ***(Sub-exponential Tail Bound for Exponential Distribution.)*** *For random variables $X_i, \ i = 1, \ldots, n$ following i.i.d exponential distributions with mean $1/\lambda$, if $n \geq 8\log(T)$, then one has:*

$$\mathbb{P}\left[\left|\frac{1}{\lambda} - \frac{1}{n}\sum_{i=1}^{n} X_i\right| \geq \frac{4}{\lambda}\sqrt{\frac{2\log(T)}{n}}\right] \leq \frac{2}{T^4}. \tag{B.2}$$

Proof of Lemma B.2.2. By Lemma B.2.1, according to the concentration inequality of sub-exponential random variables, one has

$$\mathbb{P}\left[\left|\mathbb{E}[X_i] - \frac{1}{n}\sum_{i=1}^{n} X_i\right| \geq t\right] \leq \begin{cases} 2\exp\left(-\frac{nt^2}{2\frac{4}{\lambda^2}}\right) & \text{if } 0 \leq t \leq \frac{2}{\lambda} \\ 2\exp\left(-\frac{nt}{2\frac{2}{\lambda}}\right) & \text{if } t > \frac{2}{\lambda} \end{cases}. \tag{B.3}$$

Plug in $t = \frac{4}{\lambda}\sqrt{\frac{2\log(T)}{n}}$ and $n \geq 8\log(T)$, one has $\frac{4}{\lambda}\sqrt{\frac{2\log(T)}{n}} \leq \frac{2}{\lambda}$, and thus by (B.3):

$$\begin{aligned}\mathbb{P}\left[\left|\frac{1}{\lambda} - \frac{1}{n}\sum_{i=1}^{n} X_i\right| \geq \frac{4}{\lambda}\sqrt{\frac{2\log(T)}{n}}\right] &\leq 2\exp\left(-\frac{n\frac{16}{\lambda^2}\frac{2\log(T)}{n}}{2\frac{4}{\lambda^2}}\right) \\ &= 2\exp\left(-4\log(T)\right) \\ &= \frac{2}{T^4}.\end{aligned}$$

Q.E.D.

Lemma B.2.3. ***(Sub-exponential Tail Bound for Poisson Process.)*** *For random variables $X_i,\ i = 1, \ldots, n$ following i.i.d exponential distributions with mean $1/\lambda$, if $n \geq 8\log(T)$, then one has:*

$$\mathbb{P}\left[\left|\lambda - \frac{n}{\sum_{i=1}^n X_i}\right| \geq \frac{4}{\frac{1}{n}\sum_{i=1}^n X_i}\sqrt{\frac{2\log(T)}{n}}\right] \leq \frac{2}{T^4}. \tag{B.4}$$

Proof of Lemma B.2.3. We can view $X_i,\ i = 1, \ldots, n$ as time intervals between events in a Poisson process with rate λ. Therefore, $\frac{n}{\sum_{i=1}^n X_i}$ is an empirical estimate of λ. Denote $g = \frac{n}{\sum_{i=1}^n X_i}$ and $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$, one has:

$$\begin{aligned}
&\mathbb{P}\left[\left|\lambda - \frac{n}{\sum_{i=1}^n X_i}\right| \geq \frac{4}{\frac{1}{n}\sum_{i=1}^n X_i}\sqrt{\frac{2\log(T)}{n}}\right] \\
&= \mathbb{P}\left[\left|\frac{\lambda - g}{\lambda g}\right| \geq \frac{4}{\bar{X}\lambda g}\sqrt{\frac{2\log(T)}{n}}\right] \\
&= \mathbb{P}\left[\left|\frac{1}{n}\sum_{i=1}^n X_i - \frac{1}{\lambda}\right| \geq \frac{4}{\lambda}\sqrt{\frac{2\log(T)}{n}}\right] \\
&\leq \frac{2}{T^4},
\end{aligned}$$

where the last inequality is the direct result of Lemma B.2.2.

Q.E.D.

B.3 Proofs in §4.4.1

Proof of Lemma 4.4.1. To compare random variables A_i and A_j where $i < j$ of an M/M/c queue process Q, we construct an auxiliary process $\hat{Q}$, which is the same to Q except that when the system has n customers with $j \leq n \leq c + j - i$, the service rate of $\hat{Q}$ is $n - j + i$. Similarly, we define $\hat{A}_n$ for $n = 1, \ldots, \infty$ of $\hat{Q}$. Lemma 1 in Omahen and Marathe (1978), Daley and Servi (1998) shows the recursion between A_n, for $n = 1, \ldots, \infty$. A_n can be decomposed into intervals shown in Figure B.1. For $n = 1, \ldots, \infty$, define events $\mathcal{B}_n^a$ as where a new customer arrives before any current customers leave starting from n customers in the system, and t_n^a as the time when the new customer arrives. Similarly, define events $\mathcal{B}_n^l$ as where a current customer leaves before any new customer arrive, starting from n customers in the system and define t_n^l as the time when the first current customer leaves. Therefore, for $n = 1, \ldots, \infty$, the following recursion holds:

$$\bar{A}_n = \mathbb{P}\left(\mathcal{B}_n^a\right) \cdot \left(\mathbb{E}\left[t_n^a\right] + \bar{A}_{n+1} + \bar{A}_n\right) + \mathbb{P}\left(\mathcal{B}_n^l\right) \cdot \mathbb{E}\left[t_n^l\right]. \tag{B.5}$$

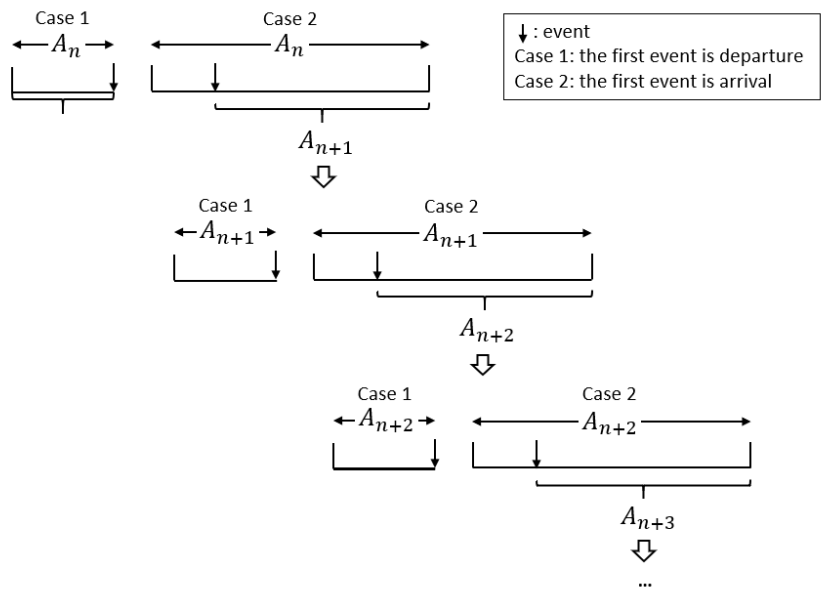


Figure B.1: Intervals within Busy Period A_n.

Comparing Q and $\hat{Q}$, we conclude that:

1. $\hat{A}_j$ and A_i have the same distribution, because they have the same arrival and departure rates in every interval in Figure B.1.

2. $\mathbb{P}(A_j \leq x) > \mathbb{P}(\hat{A}_j \leq x)$, because $\hat{A}_j$ has the same arrival rate, but a smaller (when $n < c + j - i$) or same (when $n \geq c + j - i$) departure rate in the intervals in Figure B.1.

Therefore, $\mathbb{P}(A_j \leq x) > \mathbb{P}(A_i \leq x)$. By Definition 1 in Seth and Yalonetzky (2014), one further has $\mathbb{E}_{A_j}[u(x)] < \mathbb{E}_{A_i}[u(x)]$ for all strictly increasing, continuous utility functions $u(x) : [0, \infty] \to \mathbb{R}$. This completes the proof. **Q.E.D.**

Proof of Proposition 4.4.1. For any $t \geq \tau$, one has

$$\mathbb{P}(\mathcal{A}_\tau) = \mathbb{P}(\hat{S}_\tau = S_\tau),$$

where the equality holds because once $\hat{S}_\tau = S_\tau$, then the two processes have the equal number of customers at any time thereafter and thus the two processes couple. Define $\bar{S} = \left\lfloor \frac{\log(T)\log\log(T)}{-\log\rho} \right\rfloor$, we have

$$\mathbb{P}(\hat{S}_\tau = S_\tau) \geq \mathbb{P}(\hat{S}_\tau = S_\tau | \hat{S}_0 \leq \bar{S})\mathbb{P}(\hat{S}_0 \leq \bar{S}) \geq \mathbb{P}\left(\sum_{n=1}^{\bar{S}} A_n \leq \tau\right)\mathbb{P}(\hat{S}_0 \leq \bar{S}). \tag{B.6a}$$

Define the stationary distribution for n customers in the system as $\ell_n,\ n \geq 0$, one has (by Assumption 4.2.1)

$$\mathbb{P}(S_0 \leq \bar{S}) = 1 - \frac{\ell_0 c^c}{c!}\rho^{\bar{S}+1} \geq 1 - \frac{1}{T^3}.$$

By Lemma 4.4.1, A_1 first-order dominates A_n where $n > 1$, and thus one can derive that $\mathbb{P}(\sum_{n=1}^{\bar{S}} A_1 \leq \tau)$ is a valid lower bound of $\mathbb{P}(\sum_{n=1}^{\bar{S}} A_n \leq \tau)$. Define $\bar{A}_1$ as the mean and σ_1 as the standard deviation of random variable A_1. Based on the recursive analysis of M/M/c queue (see Omahen and Marathe 1978, Daley and Servi 1998), $\bar{A}_1$ can be derived by $\bar{A}_c$, where $\bar{A}_c = \frac{1}{c\mu-\lambda}$. By Lemma B.3.1, we have

$$\mathbb{P}(\hat{S}_\tau = S_\tau) \geq \mathbb{P}\left(\sum_{n=1}^{\bar{S}} A_1 \leq \tau\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7a}$$

$$= \left(1 - \mathbb{P}\left(\sum_{n=1}^{\bar{S}} \left(A_1 - \bar{A}_1\right) > \tau - \bar{S}\bar{A}_1\right)\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7b}$$

$$= \left(1 - \mathbb{P}\left(\sum_{n=1}^{\bar{S}} \left(A_1 - \bar{A}_1\right) > \log(T)(\log\log(T))^2 - \bar{A}_1\left\lfloor\frac{\log(T)\log\log(T)}{-\log\rho}\right\rfloor\right)\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7c}$$

$$\geq \left(1 - \mathbb{P}\left(\sum_{n=1}^{\bar{S}} \left(A_1 - \bar{A}_1\right) > C_3\log(T)(\log\log(T))^{\frac{3}{2}}\right)\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7d}$$

$$\geq \left(1 - \mathbb{P}\left(\frac{1}{\bar{S}}\sum_{n=1}^{\bar{S}} \left(A_1 - \bar{A}_1\right) > C_4(\log\log(T))^{\frac{1}{2}}\right)\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7e}$$

$$\geq \left(1 - \mathbb{P}\left(\frac{1}{\bar{S}}\sum_{n=1}^{\bar{S}} \left(A_1 - \bar{A}_1\right) > 2\sigma_1\sqrt{-\log\rho}\right)\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7f}$$

$$\geq \left(1 - e^{-\log(T)\log\log(T)}\right)\left(1 - \frac{1}{T^3}\right) \tag{B.7g}$$

$$\geq \left(1 - \frac{1}{T^3}\right)^2 \geq 1 - \frac{2}{T^3}, \tag{B.7h}$$

where C_3 and C_4 are positive constants. **Q.E.D.**

Proof of Proposition 4.4.2. Without loss of generality, we consider two cases, (i) the price changes from a lower price p_l to a higher price p_h and (ii) the price changes from a higher price p_h to a lower price p_l. For each case, we further consider two sub-cases, (i) the unit service rate $\mu_{p_l} > \mu_{p_h}$ and (ii) $\mu_{p_l} \leq \mu_{p_h}$. For each case, we construct a virtual system and show that the actual probability is bounded by the probability with which the virtual system reaches the steady state, respectively. Denote the actual system as S and the virtual system as $\breve{S}$. For $t \in \{\tau, 2\tau\}$, define

events:

$$\mathcal{C}_t = \{S \text{ reaches the steady state under the new price before } t\}, \tag{B.8}$$

$$\breve{\mathcal{C}}_t = \{\breve{S} \text{ reaches the steady state under the new price before } t\}. \tag{B.9}$$

Case (1.1): from p_l to p_h and $\mu_{p_l} > \mu_{p_h}$. Without loss of generality, the system starts from $t = 0$ with S_0 customers in the system under service rate μ_{p_l}. New customers arrive under rate λ_{p_h} with an exponential service time with mean $1/\mu_{p_h}$. Consider a virtual system $\breve{S}$ starts with S_0 customers. The future customers arrive as the same process as the actual system, i.e., with rate λ_{p_h}. All the existing customers and future customers have an exponential service time with mean $1/\mu_{p_h}$. Therefore, the probability of the virtual system reaching the steady state before 2τ is the probability that an M/M/c queue under price p_h reaching the steady state with the number of initial customers as S_0. Therefore, we can compute the probability as:

$$\mathbb{P}\left(\breve{\mathcal{C}}_{2\tau}\right) \tag{B.10a}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_{2\tau} \middle| S_0 \leq 2\bar{S}_h\right) \cdot \mathbb{P}\left(S_0 \leq 2\bar{S}_h\right) \tag{B.10b}$$

$$\geq \mathbb{P}\left(\sum_{n=1}^{2\bar{S}_h} \bar{A}_{1h} \leq 2\tau\right) \cdot \mathbb{P}\left(S_0 \leq 2\bar{S}_h\right) \tag{B.10c}$$

$$\geq \left(\mathbb{P}\left(\sum_{n=1}^{\bar{S}_h} \bar{A}_{1h} \leq \tau\right)\right)^2 \cdot \mathbb{P}\left(S_0 \leq 2\bar{S}_h\right) \tag{B.10d}$$

$$\geq \left(1 - \frac{1}{T^3}\right)^2 \left(1 - \frac{\ell_{0l} c^c}{c!} \rho_l^{2\bar{S}_h+1}\right) \tag{B.10e}$$

$$\geq \left(1 - \frac{1}{T^3}\right)^2 \left(1 - \rho_l^{\frac{2}{\log \rho_h}(-\log(T)\log\log(T))}\right) \tag{B.10f}$$

$$\geq \left(1 - \frac{1}{T^3}\right)^3 \tag{B.10g}$$

$$\geq 1 - \frac{3}{T^3}, \tag{B.10h}$$

where $\bar{S}_h = \frac{\log(T)\log\log(T)}{-\log \rho_{p_h}}$ and ℓ_{0l} is the steady-state probability of the system with zero customers under price p_l. The inequality (B.10g) holds because of Assumption 4.2.2.3.

The virtual system considers a longer service time for the existing customers arrived under a lower price in the previous batch. Therefore, in probabilistic sense, the virtual system takes longer time to reach steady state:

$$\mathbb{P}(\mathcal{C}_{2\tau}) \geq \mathbb{P}(\breve{\mathcal{C}}_{2\tau}) \geq 1 - \frac{3}{T^3}. \tag{B.11}$$

Case (1.2): from p_l to p_h and $\mu_{p_l} \leq \mu_{p_h}$. Consider a virtual system $\breve{S}$ starts with S_0 customers. The future customers arrive as the process under p_l, i.e., with rate λ_{p_l}. All the existing customers and future customers have an exponential service time with mean $\frac{1}{\mu_{p_l}}$. Therefore, the virtual system is still an M/M/c queue under p_l. Denote the first time that the virtual system has zero customers as $\hat{t}$. After $\hat{t}$, the further customers arrive with rate λ_{p_h} have an exponential service time with mean $1/\mu_{p_h}$. Denote the time starting from $\hat{t}$ that the virtual system use to reach the steady state as $\breve{t}$. Therefore, the probability of the virtual system reaching the steady state under p_l before 2τ is the probability that $\hat{t} + \breve{t} \leq 2\tau$. Therefore, we can compute the probability as:

$$
\begin{aligned}
&\mathbb{P}\left(\breve{\mathcal{C}}_{2\tau}\right) && \text{(B.12a)}\\
=&\mathbb{P}\left(\breve{t} + \hat{t} \leq 2\tau\right) && \text{(B.12b)}\\
\geq&\mathbb{P}\left(\breve{t} + \hat{t} \leq 2\tau | S_0 \leq 2\bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq 2\bar{S}_l\right) && \text{(B.12c)}\\
\geq&\mathbb{P}\left(\breve{t} \leq \tau\right) \cdot \mathbb{P}\left(\hat{t} \leq \tau | S_0 \leq 2\bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq 2\bar{S}_l\right) && \text{(B.12d)}\\
\geq&\left(1 - \frac{2}{T^3}\right)^2 && \text{(B.12e)}\\
\geq&1 - \frac{4}{T^3}, && \text{(B.12f)}
\end{aligned}
$$

where (B.12e) is by Proposition 4.4.1. The virtual system considers a longer (or equal) service time for the existing customers arrived under a lower price in the previous batch. Therefore, in probabilistic sense, the virtual system takes longer (or equal) time to reach steady state:

$$
\mathbb{P}(\mathcal{C}_{2\tau}) \geq \mathbb{P}(\breve{\mathcal{C}}_{2\tau}) \geq 1 - \frac{4}{T^3}. \tag{B.13}
$$

Case (2.1): from p_h to p_l and $\mu_{p_l} > \mu_{p_h}$. Without loss of generality, the system starts from $t = 0$ with S_0 customers in the system under service rate μ_{p_h}. New customers arrive under rate $\lambda_{p_l} = \lambda_{p_h} + \Delta_\lambda$, where $\Delta_\lambda \geq 0$ by Assumption 4.2.2.1 and $\Delta_\lambda \leq \frac{\rho_{p_l}}{-3e \log p_l}$ by Assumption 4.2.2.2. We can manually split the arriving process of new customers as two independent Poisson arrival processes with rates λ_{p_h} and Δ_λ. Consider a virtual system starts with S_0 customers. The arrival processes of the new customers are the same as the actual system, i.e., λ_{p_h} and Δ_λ. The customers arrive according to the process with rate λ_{p_h} enter the system with an exponential service time with mean $1/\mu_{p_h}$. The customers arriving according to the process with rate Δ_λ are waiting in another buffer queue. Denote the first time that the virtual system (except for the buffer queue) has zero customers as $\hat{t}$ and the length of the buffer queue at this time as $\breve{S}^{\text{buffer}}_{\hat{t}}$. Starting from $\hat{t}$, the buffer queue merges with the virtual service system. All the customers in the buffer queue and further customers have an exponential service time with mean $\frac{1}{\mu_{p_l}}$. Therefore, after $\hat{t}$, the system is an M/M/c queue with balking and reneging under price p_l with an initial number of customers

as $\breve{S}_{\hat{t}}^{\text{buffer}}$. We analyze the virtual system before and after $\hat{t}$ separately. Before $\hat{t}$, the virtual system starts with a steady state under price p_h and keeps in the steady state until $\hat{t}$. As in Proposition 4.4.1, the virtual system reaches an empty state within time τ with a high probability:

$$\mathbb{P}\left(\hat{t} \leq \tau\right) \geq 1 - \frac{2}{T^3}. \tag{B.14}$$

Similarly, by Proposition 4.4.1, if the starting length of the system $\breve{S}_{\hat{t}}^{\text{buffer}}$ is less than $\frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}$, then the virtual system reaches the steady state under price p_l after τ with probability $1 - \frac{1}{T^3}$, starting at time $\hat{t}$. Therefore, we can compute the probability that the virtual queue reaches the steady state within 2τ as:

$$\mathbb{P}\left(\breve{\mathcal{C}}_{2\tau}\right) \tag{B.15a}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_{2\tau} | \hat{t} \leq \tau\right) \cdot \mathbb{P}\left(\hat{t} \leq \tau\right) \tag{B.15b}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_{2\tau} \middle| \hat{t} \leq \tau, \breve{S}_{\hat{t}}^{\text{buffer}} \leq \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}\right) \cdot \mathbb{P}\left(\breve{S}_{\hat{t}}^{\text{buffer}} \leq \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}} \middle| \hat{t} \leq \tau\right) \cdot \mathbb{P}\left(\hat{t} \leq \tau\right) \tag{B.15c}$$

$$= \left(1 - \frac{2}{T^3}\right)\left(1 - \frac{1}{T^3}\right) \cdot \mathbb{P}\left(\breve{S}_{\hat{t}}^{\text{buffer}} \leq \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}} \middle| \hat{t} \leq \tau\right) \tag{B.15d}$$

$$\geq \left(1 - \frac{3}{T^3}\right) \cdot \mathbb{P}\left(\breve{S}_{\tau}^{\text{buffer}} \leq \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}\right). \tag{B.15e}$$

The random variable $\breve{S}_{\tau}^{\text{buffer}}$ follows a Poisson distribution with mean $\tau\Delta_{p_l}$ and thus we have:

$$\mathbb{P}\left(\breve{S}_{\tau}^{\text{buffer}} \geq \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}\right) \tag{B.16a}$$

$$\leq \min_{a} \; e^{\tau\Delta_{p_l} e^{a} - \tau\Delta_{p_l} - a\frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}} \tag{B.16b}$$

$$\leq e^{-\log(T)\log\log(T)\left(\frac{1+\log(-\log \rho_{p_l}\Delta_\lambda \log\log(T))}{\log \rho_{p_l}}\right)} \tag{B.16c}$$

$$\leq e^{-\log(T)\log\log(T)} \tag{B.16d}$$

$$\leq \frac{1}{T^3}, \tag{B.16e}$$

where (B.16b) is by Chernoff bound. The inequality (B.16c) holds because $\frac{1+\log(-\log \rho_{p_l}\Delta_\lambda \log\log(T))}{\log \rho_{p_l}} \geq 1$ by Assumption 4.2.2.3. Therefore, we can future derive that

$$\mathbb{P}\left(\breve{\mathcal{C}}_{2\tau}\right) \geq \left(1 - \frac{3}{T^3}\right)\left(1 - \frac{1}{T^3}\right) \geq 1 - \frac{4}{T^3}. \tag{B.17}$$

The virtual system considers a longer service time for part of the new customers and holds the rest part of the new customers in the buffer queue even there are available resources before $\hat{t}$. Therefore, in probabilistic sense, the virtual system takes longer time to reach steady state:

$$\mathbb{P}\left(\mathcal{C}_{2\tau}\right) \geq \mathbb{P}\left(\breve{\mathcal{C}}_{2\tau}\right) \geq 1 - \frac{4}{T^3}. \tag{B.18}$$

Case (2.2): from p_h to p_l and $\mu_{p_l} \leq \mu_{p_h}$. Without loss of generality, the system starts from $t = 0$ with S_0 customers in the system under service rate μ_{p_h}. Consider a virtual system $\breve{S}$ starts with S_0 customers in the system under service rate μ_{p_l}. The future customers arrive in the process under rate λ_{p_l} with an exponential service time with mean $\frac{1}{\mu_{p_l}}$. To compare the coupling speed of the virtual queue, we consider the speed conditional on $S_0 \leq \bar{S}_l$. Consider another virtual system $\hat{S}$ starts with $\hat{S}_0$, sampled from the steady-state distribution under p_l and keeps in steady-state thereafter. Therefore, similar to Proposition 4.4.1, the probability that the actual system S and the virtual system $\breve{S}$ reach the steady-state equals to the probability that these two systems couple with the virtual system $\hat{S}$. We have shown that once they reach empty state together, then they couple with each other. For this case, we show an even stronger result that the actual system reach the steady-state under p_l within τ. Define events:

$$\breve{\mathcal{C}}_\tau^0 = \{\breve{S} \text{ reaches the empty state before } \tau\}, \tag{B.19}$$

$$\hat{\mathcal{C}}_\tau^0 = \{\hat{S} \text{ reaches the empty state before } \tau\}, \tag{B.20}$$

$$\mathcal{C}_\tau^0 = \{S \text{ reaches the empty state before } \tau\}. \tag{B.21}$$

Based on the above analysis, we can derive:

$$\mathbb{P}\left(\mathcal{C}_\tau\right) \tag{B.22a}$$

$$\geq \mathbb{P}\left(\mathcal{C}_\tau | S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq \bar{S}_l\right) \tag{B.22b}$$

$$\geq \mathbb{P}\left(\mathcal{C}_\tau^0 | S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(\hat{\mathcal{C}}_\tau^0\right) \tag{B.22c}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_\tau^0 | S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(\hat{\mathcal{C}}_\tau^0\right) \tag{B.22d}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_\tau^0 | S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(S_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(\hat{\mathcal{C}}_\tau^0 | \hat{S}_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(\hat{S}_0 \leq \bar{S}_l\right) \tag{B.22e}$$

$$\geq \mathbb{P}\left(\breve{\mathcal{C}}_\tau^0 | S_0 \leq \bar{S}_h\right) \cdot \mathbb{P}\left(S_0 \leq \bar{S}_h\right) \cdot \mathbb{P}\left(\hat{\mathcal{C}}_\tau^0 | \hat{S}_0 \leq \bar{S}_l\right) \cdot \mathbb{P}\left(\hat{S}_0 \leq \bar{S}_l\right) \tag{B.22f}$$

$$\geq \left(1 - \frac{1}{T^3}\right)^4 \tag{B.22g}$$

$$\geq 1 - \frac{4}{T^3}, \tag{B.22h}$$

where $\bar{S}_l = \frac{\log(T)\log\log(T)}{-\log \rho_{p_l}}$ and $\bar{S}_h = \frac{\log(T)\log\log(T)}{-\log \rho_{p_h}}$. The inequality (B.22f) is by Assumption 4.2.2.1 and $\mu_{p_l} < \mu_{p_h}$. The inequality (B.22g) is by the intermediate results we derive in Proposition 4.4.1. **Q.E.D.**

Lemma B.3.1. ***(Concentration Inequality for Independent Samples)*** *This lemma is similar to Lemma 11 in Chen and Shi (2019a). Let ξ_i be i.i.d. random variables with mean 0 and standard deviation σ. If the moment generating function of ξ_i around 0 is finite, i.e., there exists a constant $\delta > 0$ such that for any $s \in (-\delta, \delta)$ it holds that*

$$\mathbb{E}[e^{s\xi_i}] < \infty,$$

then

$$\mathbb{P}\left(\frac{1}{\bar{S}}\sum_{i=1}^{\bar{S}} \xi_i > 2\sigma\sqrt{-\log \rho}\right) \leq e^{-\log(T)\log\log(T)}.$$

where $\bar{S} = \left\lfloor \frac{\log(T)\log\log(T)}{-\log \rho} \right\rfloor$.

Proof of Lemma B.3.1. For s in $[-\delta, \delta]$ define

$$\Phi(s) = \log \mathbb{E}[e^{s\xi_i}].$$

For $x > 0$ and $s \in [0, \rho)$, by Markov's inequality one has

$$\mathbb{P}\left(\frac{1}{\bar{S}}\sum_{i=1}^{\bar{S}} \xi_i > x\right) \leq e^{\bar{S}(\Phi(s) - sx)}.$$

Let $s^* = \frac{x}{\sigma^2}$ and apply Taylor's expansion to the third order around 0, one has

$$\Phi(s^*) = \frac{1}{2}\sigma^2(s^*)^2 + \frac{1}{6}\Phi'''(0)(s^*)^3.$$

Therefore, one has

$$\Phi(s^*) - s^*x \leq -\frac{x^2}{2\sigma^2} + C_3\frac{x^3}{\sigma^6} \leq -\frac{x^2}{4\sigma^2}.$$

Plug in $x = 2\sigma\sqrt{-\log \rho}$ then we have

$$\mathbb{P}\left(\frac{1}{\bar{S}}\sum_{i=1}^{\bar{S}} \xi_i > 2\sigma\sqrt{-\log \rho}\right) \leq e^{\bar{S}\left(-\frac{x^2}{4\sigma^2}\right)} = e^{-\log(T)\log\log(T)}.$$

Q.E.D.

Lemma B.3.2. ***(Total Variation of M/M/c Queue)*** *For $t \geq \tau$, where $\tau = \log(T)(\log\log(T))^2$, the*

total variation of S_t and S_∞ couple is bounded as

$$\sup\{|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)|, \textit{ measurable set } \Xi \subset [0, \infty)\} \leq \frac{4}{T^3}.$$

Proof of Lemma B.3.2. As in the proof of Proposition 4.4.1, we construct a stochastic process given by $\hat{S}_0 = S_\infty$, in which $\hat{S}_t$ starts with the steady-state distribution and remains thereafter $t \geq 0$. For any $t \geq \tau$ and any measurable set $\Xi \subset [0, \infty)$, one has

$$|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)| = |\mathbb{P}(\hat{S}_t \in \Xi) - \mathbb{P}(S_t \in \Xi)| \leq \mathbb{P}(\hat{S}_\tau = S_\tau) \times 0 + \mathbb{P}(\hat{S}_\tau \neq S_\tau) \times 2,$$

where the inequality holds because once $\hat{S}_\tau = S_\tau$, then the two processes couple, and $|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)| = 0$ for any $t \geq \tau$ and any measurable set $\Xi \subset [0, \infty)$. By Proposition 4.4.1, $\mathbb{P}(\hat{S}_\tau = S_\tau) \geq 1 - \frac{2}{T^3}$, and thus

$$|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)| \leq \frac{4}{T^3}.$$

Q.E.D.

B.4 Proofs in §4.4.3

Lemma B.4.1. *For any price $p \in \mathcal{P}$,*

$$\sum_{m \in \mathcal{M}_p} I_m \tau \mathbb{E}\big[\mathbf{Rad}_m(p)\big] = O\left(\sqrt{T_p \log(T)}\right). \tag{B.23}$$

Proof of Lemma B.4.1. Plugging the definition of $\mathbf{Rad}_m(p),\ \forall m = 1, \ldots, M$, one has:

$$\begin{aligned}
\sum_{m \in \mathcal{M}_p} I_m \tau \mathbb{E}\big[\mathbf{Rad}_m(p)\big] \leq & O\left(\sqrt{\log(T)}\right) \sum_{m \in \mathcal{M}_p} I_m \tau \frac{1}{\sqrt{n_m(p)}} && \text{(B.24a)}\\
\leq & O\left(\sqrt{\log(T)}\right) \sum_{j=1}^{n_M(p)} \frac{1}{\sqrt{j}} && \text{(B.24b)}\\
\leq & O\left(\sqrt{\log(T)} \cdot \sqrt{n_M(p)}\right) && \text{(B.24c)}\\
= & O\left(\sqrt{T_p \log(T)}\right), && \text{(B.24d)}
\end{aligned}$$

where (B.24c) is by Jensen's inequality and (B.24d) is by $n_M(p) = O(T_p),\ \forall p \in \mathcal{P}$.

B.5 Proofs in §4.5.1

Lemma B.5.1. *Consider a random variable $X_i \sim SE(\nu^2, \alpha)$ and β is a non-zero scalar, then $\beta X_i \sim SE(\beta^2\nu^2, |\beta|\alpha)$.*

Lemma B.5.2. *Consider independent random variables $X_i \sim SE(\nu_i^2, \alpha_i)$ for $i = 1, \ldots, n$, then $X = \sum_{i=1}^n X_i$ follows $SE(\sum_{i=1}^n \nu_i^2, \max_i \ \alpha_i)$.*

The proofs of Lemma B.5.1 and Lemma B.5.2 are based on the definitions of SE variables and we omit the details here.

B.6 Proofs in §4.5.2

Lemma B.6.1. *Consider $\mathcal{K}_\lambda$ is a positive constant such that for any two prices p_1 and p_2, the collected arrival time observations $n_m(p_1)$ and $n_m(p_2)$ during a certain length of periods ($\geq 2\tau$) has $n_m(p_1) \leq \mathcal{K}_\lambda n_m(p_2)$ with a high probability, i.e., $1 - 1/T^4$. Constant $\mathcal{K}_\lambda$ is defined similarly for the number of service time observations. For BLinUCB algorithm, one has*

$$\sum_{m=1}^{M} n_m \cdot \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m}}\right) \leq O(\tau) + 2\mathcal{K}_\lambda d_f \log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8 l_\lambda \log(T)}\right)$$

and

$$\sum_{m=1}^{M} n_m \cdot \log\left(1 + \mu_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\mu,m}}\right) \leq O(\tau) + 2\mathcal{K}_\mu d_f \log\left(\frac{\mathcal{C}_\mu^2 O(T)}{8 l_\mu \log(T)}\right).$$

Proof of Lemma B.6.1. Without loss of generality, it is enough to prove the equation for the arrival process, i.e., the equation related to λ. By the definition of $\mathcal{M}_{\lambda,m}$, we have

$$\mathcal{M}_{\lambda,m+1} = \mathcal{M}_{\lambda,m} + n_{\lambda,m}\lambda_m^2 \mathbf{x}_m \mathbf{x}_m^T.$$

We further consider notation $\mathcal{M}_{\lambda,m,i} = \mathcal{M}_{\lambda,m} + i \cdot \lambda_m^2 \mathbf{x}_m \mathbf{x}_m^T$ and thus $\mathcal{M}_{\lambda,m,0} = \mathcal{M}_{\lambda,m}$ and $\mathcal{M}_{\lambda,m,n_m} = \mathcal{M}_{\lambda,m+1}$. Therefore, we have

$$\log\left(1 + \lambda_{p_{m+1}}^2 \|\mathbf{x}_{m+1}\|^2_{\mathcal{M}^{-1}_{\lambda,m+1}}\right) \leq \log\left(1 + \lambda_{p_{m+1}}^2 \|\mathbf{x}_{m+1}\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right), \ \forall i = 1, \ldots, n_{\lambda,m}.$$

By the theory of projection and by the definition of $\mathcal{M}_{\lambda,m,i}$, we further have

$$\log\left(1 + \lambda_{p_{m+1}}^2 \|\mathbf{x}_{m+1}\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \leq \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right), \ \forall i = 1, \ldots, n_{\lambda,m}.$$

The number of observations in one batch is in the same order as the length of the batch, i.e, $n_m = O(I_m\tau)$. Recall the batch size is $I_m\tau = 2^m\tau$, thus with a high probability that $n_m \leq 2\mathcal{K}_\lambda n_{m-1}$.

Further we can derive the following bound for $m = 2, \ldots, M$:

$$
\begin{aligned}
& n_m \cdot \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m}}\right) \\
\leq & 2 \sum_{i=1}^{n_m/2} \log\left(1 + \lambda_{p_{m-1}}^2 \|\mathbf{x}_{m-1}\|^2_{\mathcal{M}^{-1}_{\lambda,m-1,i}}\right) \\
\leq & 2\mathcal{K}_\lambda \sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_{m-1}}^2 \|\mathbf{x}_{m-1}\|^2_{\mathcal{M}^{-1}_{\lambda,m-1,i}}\right).
\end{aligned}
$$

Sum over batches $m = 1, \ldots, M$, we can reach the following result:

$$
\begin{aligned}
& \sum_{m=1}^{M} n_m \cdot \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m}}\right) && \text{(B.25a)} \\
\leq & O(\tau) + 2\mathcal{K}_\lambda \sum_{m=2}^{M} \sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_{m-1}}^2 \|\mathbf{x}_{m-1}\|^2_{\mathcal{M}^{-1}_{\lambda,m-1,i}}\right) && \text{(B.25b)} \\
\leq & O(\tau) + 2\mathcal{K}_\lambda d_f \log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8 l_\lambda \log(T)}\right). && \text{(B.25c)}
\end{aligned}
$$

where (B.25c) is by Lemma B.6.2.

Q.E.D.

Lemma B.6.2. *The feature vector of the price selected in batch m is denoted by $\mathbf{x}_m$, the number of arrival time observations of this price is denoted by n_m, and the number of service time observations of this price is denoted by n_m^s. Then*

$$
\sum_{m=1}^{M-1} \sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \leq d_f \log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8 l_\lambda \log(T)}\right).
$$

and

$$
\sum_{m=1}^{M-1} \sum_{i=1}^{n_{m-1}} \log\left(1 + \mu_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\mu,m,i}}\right) \leq d_f \log\left(\frac{\mathcal{C}_\mu^2 O(T)}{8 l_\mu \log(T)}\right).
$$

Proof of Lemma B.6.2. The proof follows similar ideas of the proof of Proposition 3 in Filippi et al. (2010). Without loss of generality, we firstly prove for arrival processes, i.e., the equation related to λ. By the definition of $\mathcal{M}_{\lambda,m,i}$ for $i = 0, 1, \ldots, n_m$, we have

$$
\mathcal{M}_{\lambda,m,i+1} = \mathcal{M}_{\lambda,m,i} + \lambda_m^2 \mathbf{x}_m \mathbf{x}_m^T, \ \forall i = 1, \ldots, n_m
$$

and two boundary equations, $\mathcal{M}_{\lambda,m,0} = \mathcal{M}_{\lambda,m}$ and $\mathcal{M}_{\lambda,m,n_m} = \mathcal{M}_{\lambda,m+1}$. Therefore, we can

compute:

$$\det(\mathcal{M}_{\lambda,m+1}) = \det\left(\mathcal{M}_{\lambda,m,n_m-1}(I + \lambda_m \mathcal{M}_{\lambda,m,n_m-1}^{-1/2}\mathbf{x}_m(\lambda_m \mathcal{M}_{\lambda,m,n_m-1}^{-1/2}\mathbf{x}_m)^T)\right) \tag{B.26a}$$

$$= \det\left(\mathcal{M}_{\lambda,m,n_m-1}\right) \cdot \det\left(I + \lambda_m \mathcal{M}_{\lambda,m,n_m-1}^{-1/2}\mathbf{x}_m(\lambda_m \mathcal{M}_{\lambda,m,n_m-1}^{-1/2}\mathbf{x}_m)^T\right) \tag{B.26b}$$

$$= \det\left(\mathcal{M}_{\lambda,m,n_m-1}\right) \cdot \left(1 + \lambda_m^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,n_m-1}}\right) \tag{B.26c}$$

$$= \det\left(\mathcal{M}_{\lambda,m}\right) \prod_{i=0}^{n_m-1}\left(1 + \lambda_m^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \tag{B.26d}$$

$$= \det\left(\mathcal{M}_{\lambda,1}\right) \prod_{k=1}^{m}\prod_{i=0}^{n_k-1}\left(1 + \lambda_k^2 \|\mathbf{x}_k\|^2_{\mathcal{M}^{-1}_{\lambda,k,i}}\right), \tag{B.26e}$$

where (B.26c) follows the fact that $1 + \lambda_m^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m}}$ is an eigenvalue of the matrix $I + \lambda_m \mathcal{M}_{\lambda,m}^{-1/2}\mathbf{x}_m(\lambda_m \mathcal{M}_{\lambda,m}^{-1/2}\mathbf{x}_m)^T$ and that all the other eigenvalues are equal to 1.

Consider the left-hand-side of the equation in the lemma, we have

$$\begin{aligned}
&\sum_{m=1}^{M-1}\sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \\
<&\sum_{m=1}^{M}\sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \\
\leq&\log \prod_{m=1}^{M}\prod_{i=0}^{n_m-1}\left(1 + \lambda_m^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \\
=&\log\left(\frac{\det(\mathcal{M}_{\lambda,M})}{\det(\mathcal{M}_{\lambda,1})}\right).
\end{aligned}$$

Denote the collected arrival time observations during the Warm-up Phase as n_w. Note that the trace of $\mathcal{M}_{m+1}$ is upper bounded by $\mathcal{C}_\lambda^2(n_w + \sum_{k=1}^m n_k)$ according to Assumption 4.5.1. Then, since the trace of the positive definite matrix $\mathcal{M}_{m+1}$ is equal to the sum of its eigenvalues and $\det(\mathcal{M}_{m+1})$ is the product of its eigenvalues, we have $\det(\mathcal{M}_{m+1}) \leq (\mathcal{C}_\lambda^2(n_w + \sum_{k=1}^m n_k))^{d_f}$. In addition, $\det(\mathcal{M}_{\lambda,1}) \geq (8\log(T) \cdot l_\lambda)^{d_f}$ by Assumption 4.5.2 and the algorithmic design that each price $p \in \mathcal{P}_b$ has more than $8\log(T)$ number of observations during the Warm-up phase. Thus

$$\sum_{m=1}^{M-1}\sum_{i=1}^{n_{m-1}} \log\left(1 + \lambda_{p_m}^2 \|\mathbf{x}_m\|^2_{\mathcal{M}^{-1}_{\lambda,m,i}}\right) \leq d_f \log\left(\frac{\mathcal{C}_\lambda^2 \sum_{m=1}^M n_m}{8 l_\lambda \log(T)}\right) = d_f \log\left(\frac{\mathcal{C}_\lambda^2 O(T)}{8 l_\lambda \log(T)}\right).$$

Similarly, we can prove the part for the service process, i.e., the equation related to μ.

Q.E.D.

B.7 Proofs in §4.6.2

Proof of Proposition 4.6.2. The proof is similar to the proof of Proposition 4.3.1. We present the intermediate results in Lemma B.7.1 to B.7.3.

We have

$$\begin{aligned}&\mathbb{P}\left(\left|\left(\frac{1}{\mu_p}\left(\lambda_p-(\nu_p+(1-\eta_p)\lambda_p)\ell_p^{br}\right)\right)-\left(\bar{g}_m(p)\left(\breve{d}_m(p)-\left(\breve{h}_m(p)+\frac{n_m^b(p)}{n_m^c(p)}\breve{d}_m(p)\right)\hat{\ell}_p^{br}\right)\right)\right|\le\Gamma\right)\\ =&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p}-\frac{\nu_p}{\mu_p}\ell_p^{br}-(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p}-\breve{d}_m(p)\bar{g}_m(p)+\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}+\frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|\le\Gamma\right).\end{aligned}\tag{B.27}$$

By plugging in the value of

$$\Gamma=\frac{4(y_{\max}+r_{\max})}{T^3}+\left(8r_{\max}+4y_{\max}+9\breve{h}_m(p)\bar{g}_m(p)+4\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^c(p),n_m^r(p)\}}},$$

we have

$$\begin{aligned}&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p}-\frac{\nu_p}{\mu_p}\ell_p^{br}-(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p}-\breve{d}_m(p)\bar{g}_m(p)+\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}+\frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|\le\Gamma\right)\\ \ge&\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p}-\breve{d}_m(p)\bar{g}_m(p)\right|\le 4\left(r_{\max}+\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)\\ &\cdot\mathbb{P}\left(\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\frac{4y_{\max}}{T^3}+4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\\ &\cdot\mathbb{P}\left(\left|(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p}-\frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|\le\frac{4r_{\max}}{T^3}+\left(4r_{\max}+5\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^c(p)\}}}\right).\end{aligned}\tag{B.28}$$

By Proposition 4.3.1, we have

$$\mathbb{P}\left(\left|\frac{\lambda_p}{\mu_p}-\breve{d}_m(p)\bar{g}_m(p)\right|\le 4\left(r_{\max}+\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)\le 1-\frac{4}{T^4}.$$

By Lemma B.7.1, we have

$$\mathbb{P}\left(\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\le\frac{4y_{\max}}{T^3}+4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\ge 1-\frac{3}{T^3}.$$

By Lemma B.7.3, we can further derive that

$$\begin{aligned}
&\mathbb{P}\left(\left|\left(\frac{1}{c\mu_p}\left(\lambda_p-(\nu_p+(1-\eta_p)\lambda_p)\ell_p^{br}\right)\right)-\left(\frac{\bar{g}_m(p)}{c}\left(\breve{d}_m(p)-(\breve{h}_m(p)+\frac{n_m^b(p)}{n_m^c(p)}\breve{d}_m(p))\hat{\ell}_p^{br}\right)\right)\right|\leq\Gamma\right)\\
\geq&\left(1-\frac{4}{T^4}\right)\left(1-\frac{3}{T^3}\right)\left(1-\frac{4}{T^3}\right)\\
\geq&1-\frac{8}{T^3}.
\end{aligned}\tag{B.29}$$

Q.E.D.

Lemma B.7.1. *If the number of the observed service time intervals $n_m^s(p)\geq 8\log(T)$ and the number of observed reneging time intervals $n_m^r(p)\geq 8\log(T)$, then, one has*

$$\mathbb{P}\left(\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\leq\frac{4y_{\max}}{T^3}+4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\geq 1-\frac{3}{T^3},$$

where $y_{\max}\geq\max_{p\in\mathcal{P}}\nu_p/\mu_p$.

Proof of Lemma B.7.1. First, we have

$$\begin{aligned}
&\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\\
=&\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\frac{\nu_p}{\mu_p}\hat{\ell}_p^{br}+\frac{\nu_p}{\mu_p}\hat{\ell}_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\\
\leq&\frac{\nu_p}{\mu_p}\left|\ell_p^{br}-\hat{\ell}_p^{br}\right|+\hat{\ell}_p^{br}\left|\frac{\nu_p}{\mu_p}-\breve{h}_m(p)\bar{g}_m(p)\right|.
\end{aligned}\tag{B.30}$$

Define event $\mathcal{G}$ as the queue is in steady state after the first 2τ time periods, and define the complementary event as $\bar{\mathcal{G}}$. Plug in $y_{\max}\geq\max_{p\in\mathcal{P}}\nu_p/\mu_p$, by Proposition 4.4.1 and Lemma B.8.1 and , we have

$$\begin{aligned}
&\mathbb{P}\left(\frac{\nu_p}{\mu_p}|\ell_p^{br}-\hat{\ell}_p^{br}|\leq\frac{4y_{\max}}{T^3}\right)\\
\geq&\mathbb{P}\left(\frac{\nu_p}{\mu_p}|\ell_p^{br}-\hat{\ell}_p^{br}|\leq\frac{4\nu_p}{\mu_pT^3}\right)\\
=&\mathbb{P}\left(\left|\ell_p^{br}-\hat{\ell}_p^{br}\right|\leq\frac{4}{T^3}\middle|\mathcal{G}\right)\cdot\mathbb{P}\left(\mathcal{G}\right)+\mathbb{P}\left(|\ell_p^{br}-\hat{\ell}_p^{br}|\leq\frac{4}{T^3}\middle|\bar{\mathcal{G}}\right)\cdot\mathbb{P}(\bar{\mathcal{G}})\\
\geq&\mathbb{P}\left(|\ell_p^{br}-\hat{\ell}_p^{br}|\leq\frac{4}{T^3}\middle|\mathcal{G}\right)\cdot\mathbb{P}(\mathcal{G})\\
\geq&1-\frac{2}{T^3}.
\end{aligned}\tag{B.31}$$

We also have:

$$\mathbb{P}\left(\hat{\ell}_p^{br}\left|\frac{\nu_p}{\mu_p}-\breve{h}_m(p)\bar{g}_m(p)\right|\leq 4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right) \tag{B.32a}$$

$$\geq\mathbb{P}\left(\left|\frac{\nu_p}{\mu_p}-\breve{h}_m(p)\cdot\bar{g}_m(p)\right|\leq 4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right) \tag{B.32b}$$

$$\geq 1-\frac{4}{T^4}, \tag{B.32c}$$

where the inequality (B.32b) is by $0\leq\hat{\ell}_p^{br}\leq 1$ and the inequality (B.32c) is by Proposition 4.3.1.

To conclude, we have

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{\nu_p}{\mu_p}\ell_p^{br}-\breve{h}_m(p)\bar{g}_m(p)\hat{\ell}_p^{br}\right|\leq\frac{4y_{\max}}{T^3}+4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\\
=&\mathbb{P}\left(\frac{\nu_p}{\mu_p}\left|\ell_p^{br}-\hat{\ell}_p^{br}\right|+\hat{\ell}_p^{br}\left|\frac{\nu_p}{\mu_p}-\breve{h}_m(p)\bar{g}_m(p)\right|\leq\frac{4y_{\max}}{T^3}+4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\\
\geq&\mathbb{P}\left(\frac{\nu_p}{\mu_p}\left|\ell_p^{br}-\hat{\ell}_p^{br}\right|\leq\frac{4y_{\max}}{T^3}\right)\cdot\mathbb{P}\left(\hat{\ell}_p^{br}\left|\frac{\nu_p}{\mu_p}-\breve{h}_m(p)\bar{g}_m(p)\right|\leq 4\left(y_{\max}+\breve{h}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^r(p)\}}}\right)\\
\geq&\left(1-\frac{2}{T^3}\right)\left(1-\frac{4}{T^4}\right)\\
\geq&1-\frac{3}{T^3}.
\end{aligned}$$

Q.E.D.

Lemma B.7.2. *(Concentration inequality of balking probability.)*

$$\mathbb{P}\left(\left|\frac{n_m^b(p)}{n_m^c(p)}-(1-\eta_p)\right|\leq\sqrt{\frac{2\log(T)}{n_m^c(p)}}\right)\geq 1-\frac{2}{T^4}.$$

Proof of Lemma B.7.2. Denote $X_1,\ X_2,\ \ldots,\ X_{n_m^c}$ as random variables such that $X_i=1$ if the customer balks and $X_i=0$ otherwise. Therefore, by Hoeffding's inequality we have

$$\mathbb{P}\left(\left|\sum_{i=1}^{n_m^c(p)}X_i-\mathbb{E}\left[\sum_{i=1}^{n_m^c(p)}X_i\right]\right|\leq a\right)\geq 1-2\exp\left(-\frac{2a^2}{n_m^c(p)}\right).$$

By letting $a = \sqrt{2\log(T) n_m^c(p)}$, we have:

$$\begin{aligned}
&\mathbb{P}\left(\left|\frac{\sum_{i=1}^{n_m^c(p)} X_i}{n_m^c(p)} - \frac{\mathbb{E}[\sum_{i=1}^{n_m^c(p)} X_i]}{n_m^c(p)}\right| \leq \frac{a}{n_m^c(p)}\right) \geq 1 - 2\exp\left(-\frac{2a^2}{n_m^c(p)}\right) \\
\Rightarrow &\mathbb{P}\left(\left|\frac{n_m^b(p)}{n_m^c(p)} - (1-\eta_p)\right| \leq \frac{a}{n_m^c(p)}\right) \geq 1 - 2\exp\left(-\frac{2a^2}{n_m^c(p)}\right) \\
\Rightarrow &\mathbb{P}\left(\left|\frac{n_m^b(p)}{n_m^c(p)} - (1-\eta_p)\right| \leq \sqrt{\frac{2\log(T)}{n_m^c(p)}}\right) \geq 1 - \frac{2}{T^4}.
\end{aligned}$$

Q.E.D.

Lemma B.7.3. *If $n_m^s(p) \geq 8\log(T)$ and $n_m^c(p) \geq 8\log(T)$, then, one has*

$$\mathbb{P}\left(\left|(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p} - \frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| \leq \frac{4r_{\max}}{T^3} + \left(4r_{\max} + 5\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p), n_m^c(p)\}}}\right) \geq 1 - \frac{4}{T^3}.$$

Proof of Lemma B.7.3. First, we can derive that

$$\begin{aligned}
&\left|(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p} - \frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| \\
=&\left|(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p} - (1-\eta_p)\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p) + (1-\eta_p)\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p) - \frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| \\
\leq&(1-\eta_p)\cdot\left|\ell_p^{br}\frac{\lambda_p}{\mu_p} - \hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| + \hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\cdot\left|(1-\eta_p) - \frac{n_m^b(p)}{n_m^c(p)}\right| \\
\leq&\left|\ell_p^{br}\frac{\lambda_p}{\mu_p} - \hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| + \breve{d}_m(p)\bar{g}_m(p)\cdot\left|(1-\eta_p) - \frac{n_m^b(p)}{n_m^c(p)}\right|.
\end{aligned} \tag{B.33}$$

Similar to Lemma B.7.1, we have

$$\mathbb{P}\left(\left|\ell_p^{br}\frac{\lambda_p}{\mu_p} - \hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right| \leq \frac{4r_{\max}}{T^3} + 4\left(r_{\max} + \breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right) \geq 1 - \frac{3}{T^3}. \tag{B.34}$$

By Lemma B.7.2, we have

$$\begin{aligned}
&\mathbb{P}\left(\breve{d}_m(p)\bar{g}_m(p)\cdot\left|(1-\eta_p) - \frac{n_m^b(p)}{n_m^c(p)}\right| \leq \breve{d}_m(p)\bar{g}_m(p)\sqrt{\frac{2\log(T)}{n_m^c(p)}}\right) \\
=&\mathbb{P}\left(\left|(1-\eta_p) - \frac{n_m^b(p)}{n_m^c(p)}\right| \leq \sqrt{\frac{2\log(T)}{n_m^c(p)}}\right) \\
\geq&1 - \frac{2}{T^4}.
\end{aligned} \tag{B.35}$$

Therefore, we have

$$
\begin{aligned}
&\mathbb{P}\left(\left|(1-\eta_p)\ell_p^{br}\frac{\lambda_p}{\mu_p}-\frac{n_m^b(p)}{n_m^c(p)}\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|\le\frac{4r_{\max}}{T^3}+\left(4r_{\max}+5\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^c(p)\}}}\right)\\
=&\mathbb{P}\left(\left|\ell_p^{br}\frac{\lambda_p}{\mu_p}-\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|+\breve{d}_m(p)\bar{g}_m(p)\cdot\left|(1-\eta_p)-\frac{n_m^b(p)}{n_m^c(p)}\right|\right.\\
&\qquad\left.\le\frac{4r_{\max}}{T^3}+\left(4r_{\max}+5\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{\min\{n_m^s(p),n_m^c(p)\}}}\right)\\
\ge&\mathbb{P}\left(\left|\ell_p^{br}\frac{\lambda_p}{\mu_p}-\hat{\ell}_p^{br}\breve{d}_m(p)\bar{g}_m(p)\right|\le\frac{4r_{\max}}{T^3}+4\left(r_{\max}+\breve{d}_m(p)\bar{g}_m(p)\right)\sqrt{\frac{2\log(T)}{n_m^s(p)}}\right)\\
&\qquad+\mathbb{P}\left(\breve{d}_m(p)\bar{g}_m(p)\cdot\left|(1-\eta_p)-\frac{n_m^b(p)}{n_m^c(p)}\right|\le\breve{d}_m(p)\bar{g}_m(p)\sqrt{\frac{2\log(T)}{n_m^c(p)}}\right)\\
\ge&\left(1-\frac{3}{T^3}\right)\left(1-\frac{2}{T^4}\right)\\
\ge&1-\frac{4}{T^3}.
\end{aligned}
$$

Q.E.D.

B.8 Proofs in §4.6.3

Proof of Lemma 4.6.1. Based on the proof of Lemma 4.4.1, we skip the step for constructing an auxiliary queue but directly analyze the service and arrival rate of each interval. Please note that balking and reneging can be included as part of both arrival or departure rate. In the following proofs, we include them in the arrival rate.

Consider an M/M/c queue Q with balking and reneging where Assumption 4.6.2.1 and Assumption 4.6.2.2 hold. Similarly, we can analyze the intervals included in A_i and A_j:

1. All intervals in A_i has the same or a smaller service rate compared with the intervals sit in the same position in the recursion illustration of A_j, because the system has c servers and thus the former always has the less or same amount of customers under service.

2. All intervals in A_i has the same or a larger average arrival rate compared with the intervals sit in the same position in the recursion illustration of A_j according to Assumption 4.6.2.1 and Assumption 4.6.2.2.

Therefore, $\mathbb{P}(A_j \le x) > \mathbb{P}(A_i \le x)$ and A_i first-order dominates A_j. **Q.E.D.**

Proof of Proposition 4.6.3. For any $t \ge \tau$, one has

$$\mathbb{P}(\mathcal{A}_\tau) = \mathbb{P}\left(\hat{S}_\tau = S_\tau\right),$$

where the equality holds because once $\hat{S}_\tau = S_\tau$, then the two processes couple. Define $\bar{S} = \left\lfloor \frac{\log(T)\log\log(T)}{-\log\rho} \right\rfloor$. By Lemma 4.6.1, A_1 first-order dominates A_n where $n > 1$, and thus one can derive that $\mathbb{P}(\sum_{n=1}^{\bar{S}} A_1 \leq \tau)$ is a valid lower bound of $\mathbb{P}(\sum_{n=1}^{\bar{S}} A_n \leq \tau)$. Therefore, we have

$$\mathbb{P}\left(\hat{S}_\tau = S_\tau\right) \tag{B.36a}$$

$$\geq \mathbb{P}\left(\hat{S}_\tau = S_\tau | \hat{S}_0 \leq \bar{S}\right) \mathbb{P}\left(\hat{S}_0 \leq \bar{S}\right) \tag{B.36b}$$

$$= \mathbb{P}\left(\sum_{n=1}^{\bar{S}} A_n \leq \tau\right) \mathbb{P}\left(\hat{S}_0 \leq \bar{S}\right) \tag{B.36c}$$

$$\geq \mathbb{P}\left(\sum_{n=1}^{\bar{S}} A_1 \leq \tau\right) \mathbb{P}\left(\hat{S}_0 \leq \bar{S}\right). \tag{B.36d}$$

By Lemma B.8.5 and Lemma B.8.4, (B.36d) can be bounded by

$$\mathbb{P}\left(\sum_{n=1}^{\bar{S}} A_1 \leq \tau\right) \mathbb{P}\left(S_0 \leq \bar{S}\right) \tag{B.37a}$$

$$\geq \mathbb{P}\left(\sum_{n=1}^{\bar{S}} \tilde{A}_1 \leq \tau\right) \mathbb{P}\left(\tilde{N} \leq \bar{S}\right). \tag{B.37b}$$

In the proof of Proposition 4.4.1, we show that (B.37b) is bounded blow by $1 - \frac{2}{T^3}$. **Q.E.D.**

Proof of Proposition 4.6.4. The proof is similar to the proof of Proposition 4.4.2 and we omit the details here. **Q.E.D.**

Lemma B.8.1. ***(Total Variation of M/M/c Queue with Balking and Reneging)*** *For $t \geq \tau$, where $\tau = \log(T)(\log\log(T))^2$, the total variation of S_t and S_∞ is bounded as*

$$\sup\left\{\left|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)\right|, \textit{ measurable set } \Xi \subset [0,\infty)\right\} \leq \frac{4}{T^3}.$$

Proof of Lemma B.8.1. The proof here is similar to the proof of Lemma B.3.2. Based on Proposition 4.6.3, we have for any $t \geq \tau$ and any measurable set $\Xi \subset [0,\infty)$:

$$|\mathbb{P}(S_t \in \Xi) - \mathbb{P}(S_\infty \in \Xi)| = \left|\mathbb{P}\left(\hat{S}_t \in \Xi\right) - \mathbb{P}(S_t \in \Xi)\right| \leq \mathbb{P}\left(\hat{S}_\tau = S_\tau\right) \times 0 + \mathbb{P}\left(\hat{S}_\tau \neq S_\tau\right) \times 2 \leq \frac{4}{T^3}.$$

Q.E.D.

Lemma B.8.2. ***(First-order Stochastic Dominance between A_i and A_j in M/M/c Queue with Balking)*** *If Assumption 4.6.2.1 holds, then for random variables A_i and A_j where $i < j$, A_i first-order dominates A_j. Similar to Lemma 4.4.1, one has $\mathbb{P}(A_j \leq x) \geq \mathbb{P}(A_i \leq x)$ and $\mathbb{E}_{A_j}[u(x)] \leq \mathbb{E}_{A_i}[u(x)]$ for all strictly increasing, continuous utility functions $u(x) : [0,\infty] \to \mathbb{R}$.*

Proof of Lemma B.8.2. Consider an M/M/c queue Q with balking where Assumption 4.6.2.1 holds. The following analysis is also based on the arguments in Lemma 1 in Omahen and Marathe (1978) and this recursion relationship is illustrated in Figure B.1. We can analyze the intervals included in A_i and A_j:

1. All intervals in A_i has the same or a smaller service rate compared with the intervals sit in the same position in the recursion illustration of A_j, because the system has c servers and thus the former always has the less or same amount of customers under service.

2. All intervals in A_i has the same or a larger average arrival rate compared with the intervals sit in the same position in the recursion illustration of A_j according to Assumption 4.6.2.1.

Therefore, $\mathbb{P}(A_j \leq x) > \mathbb{P}(A_i \leq x)$ and A_i first-order dominates A_j. **Q.E.D.**

Lemma B.8.3. ***(First-order Stochastic Dominance between A_i and A_j in M/M/c Queue with Reneging)*** *If Assumption 4.6.2.2 holds, then for random variables A_i and A_j where $i < j$, A_i first-order dominates A_j. Similarly, one has $\mathbb{P}(A_j \leq x) \geq \mathbb{P}(A_i \leq x)$ and $\mathbb{E}_{A_j}[u(x)] \leq \mathbb{E}_{A_i}[u(x)]$ for all strictly increasing, continuous utility functions $u(x) : [0, \infty] \to \mathbb{R}$.*

Proof of Lemma B.8.3. Consider an M/M/c queue Q with reneging where Assumption 4.6.2.2 holds. Our analysis is based on the arguments in Lemma 1 in Omahen and Marathe (1978) and this recursion relationship is illustrated in Figure B.1. We analyze the intervals included in A_i and A_j:

1. All intervals in A_i has the same or a smaller service rate compared with the intervals sit in the same position in the recursion illustration of A_j, because the system has c servers and thus the former always has the less or same amount of customers under service.

2. All intervals in A_i has the same or a larger average arrival rate compared with the intervals sit in the same position in the recursion illustration of A_j according to Assumption 4.6.2.2.

Therefore, $\mathbb{P}(A_j \leq x) > \mathbb{P}(A_i \leq x)$ and A_i first-order dominates A_j. **Q.E.D.**

We further establish the first-order stochastic dominance between the standard M/M/c queue and the M/M/c queue with Balking and Reneging under Assumptions 4.6.2.1 and 4.6.2.2. Consider the standard M/M/c queue and the M/M/c queue with Balking and Reneging under the same customer arrival and service rate. Denote the number of customers of M/M/c queue with Balking and Reneging in steady state as N and denote that of standard M/M/c queue as $\tilde{N}$. Denote the busy period of the standard M/M/c queue as $\tilde{A}_i$.

Lemma B.8.4. ***(First-order Stochastic Dominance between $\tilde{A}_i$ and A_i.)*** *$\tilde{A}_i$ first-order dominates A_i. Thus, one has $\mathbb{P}(A_i \leq x) \geq \mathbb{P}(\tilde{A}_i \leq x)$ and $\mathbb{E}_{A_i}[u(n)] \leq \mathbb{E}_{\tilde{A}_i}[u(x)]$ for all strictly increasing, continuous utility functions $u(x) : [0, \infty] \to \mathbb{R}$.*

Proof of Lemma B.8.4. Similarly, we can analyze the intervals included in A_i and $\tilde{A}_i$:

1. All intervals in A_i has the same service rate compared with those of $\tilde{A}_i$.

2. All intervals in A_i has the same or a smaller average arrival rate compared with those of $\tilde{A}_i$.

Therefore, $\mathbb{P}(A_i \leq x) > \mathbb{P}(\tilde{A}_i \leq x)$ and $\tilde{A}_i$ first-order dominates A_i. **Q.E.D.**

Lemma B.8.5. ***(First-order Stochastic Dominance between $\tilde{N}$ and N.)*** *$\tilde{N}$ first-order dominates N. Thus, one has $\mathbb{P}(N \leq n) \geq \mathbb{P}(\tilde{N} \leq n)$ and $\mathbb{E}_N[u(n)] \leq \mathbb{E}_{\tilde{N}}[u(x)]$ for all strictly increasing, continuous utility functions $u(n) : [0, 1, 2, \ldots] \to \mathbb{R}$.*

Proof of Lemma B.8.5. The proof here is directly originated from the overall arrival and service rates of these two queues. The queue with balking and reneging always has a smaller arrival rate and thus $\mathbb{P}(N \leq n) \geq \mathbb{P}(\tilde{N} \leq n)$. From another perspective, to avoid the effects of the number of customers in the system on service rate, we can view balking and reneging as part of the departure rate. Thus, the queue with balking and reneging has the same arrival but larger departure rate, and further one has $\mathbb{P}(N \leq n) \geq \mathbb{P}(\tilde{N} \leq n)$. **Q.E.D.**

Proof of Theorem 4.6.1. Similar to Theorem 4.4.1, we bound the relaxed regret during the Warm-up Phase and the relaxed regret of suboptimality and nonstationarity in the Learning Phase separately.

Regret of the Warm-up Phase. The difference between $J^*_{\text{Warm-up}}$ and $J^{\text{BUCB}}_{\text{Warm-up}}$ is at most linear on the length of the Warm-up Phase, which is $O(P\log(T))$. Therefore we can directly have

$$\overline{\text{Regret}}^{\text{IND}}_{\text{Warm-up}}(\pi_{\text{BUCB}}, T) = O(P\log(T)). \tag{B.38}$$

Regret of suboptimality during the Learning Phase. Denote the offered price of batch m in BUCB is p_m. For price p, denote the difference of revenue rate with the state-independent optimal price as

$$\begin{aligned}
\Delta(p) &= r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - r(\lambda_p, \mu_p, \nu_p, \eta_p, p) \\
&= \big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U^E_m(p)\big) + (U^E_m(p) - U^E_m(p)) + (L^E_m(p) - r(\lambda_p, \mu_p, \nu_p, \eta_p, p) \\
&\leq \big(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U^E_m(\tilde{p})\big) + (U^E_m(p) - U^E_m(p)) + (L^E_m(p) - r(\lambda_p, \mu_p, \nu_p, \eta_p, p)
\end{aligned} \tag{B.39}$$

following a same logic as that in (4.14).

Regret of nonstationarity during the Learning Phase. For batch m and the offered price p_m, we derive the total collected revenue as

$$J_m(p_m) = J_{m,0\sim 2\tau}(p_m) + J_{m,2\tau\sim I_m\tau}(p_m),$$

where $J_{m,0\sim 2\tau}(p_m)$ denotes the revenue collected in the first 2τ periods and $J_{m,2\tau\sim I_m\tau}(p_m)$ denotes the revenue collected from $2\tau+1$ period to the end of batch m. We keep the notation of the event $\mathcal{C}_{2\tau}$ to be the case that the system reaches the steady state under the new price and define $\bar{\mathcal{C}}_{2\tau}$ as the complementary event of $\mathcal{C}_{2\tau}$. By Proposition 4.4.2, one has $\mathbb{P}[\mathcal{C}_{2\tau}] \geq 1 - \frac{4}{T^3}$. Consequently, we denote $J_{m,\mathcal{C}_{2\tau}}(p_m)$ as the revenue of $J_{m,2\tau\sim I_m\tau}(p_m)$ conditional on $\mathcal{C}_{2\tau}$ and $J_{m,\bar{\mathcal{C}}_{2\tau}}(p_m)$ otherwise. Therefore, we have

$$\begin{aligned}
\mathbb{E}[J_m(p_m)] &= \mathbb{E}[J_{m,0\sim 2\tau}(p_m) + J_{m,\mathcal{C}_{2\tau}}(p_m)\mathbb{P}(\mathcal{C}_{2\tau}) + J_{m,\bar{\mathcal{C}}_{2\tau}}(p_m)(1-\mathbb{P}(\mathcal{C}_{2\tau}))] \\
&\geq O(\tau) + \left(\frac{p_m}{\mu_{p_m}}\left(\lambda_{p_m} - (\nu_{p_m} + (1-\eta_{p_m})\lambda_{p_m})\ell^{br}_{p_m}\right)\right)(I_m\tau - 2\tau)\left(1 - \frac{4}{T^3}\right) + O\left(\frac{I_m\tau - 2\tau}{T^3}\right) \\
&= \left(\frac{p_m}{\mu_{p_m}}\left(\lambda_{p_m} - (\nu_{p_m} + (1-\eta_{p_m})\lambda_{p_m})\ell^{br}_{p_m}\right)\right) I_m\tau - O(\tau) \\
&= r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m) I_m\tau - O(\tau).
\end{aligned}$$

Then the cumulative regret by the end of the Learning Phase

$$\overline{\text{Regret}}^{\text{IND}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) = J^{\text{IND}} - \sum_{m=1}^{M} J_m(p_m) \tag{B.40a}$$

$$= \mathbb{E}\left[\sum_{m=1}^{M}\left(r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) I_m\tau - J_m(p_m)\right)\right] \tag{B.40b}$$

$$\leq \mathbb{E}\left[\sum_{m=1}^{M}\left(\Delta(p_m) I_m\tau + O(\tau)\right)\right]. \tag{B.40c}$$

Plugging in (B.39), we have

$$\begin{aligned}
\sum_{m=1}^{M}\mathbb{E}\Big[\Delta(p_m) I_m\tau\Big] &= \sum_{m=1}^{M}\mathbb{E}\Big[\left(r(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U^E_m(\tilde{p})\right)\Big] I_m\tau + \sum_{m=1}^{M}\mathbb{E}\Big[(U^E_m(p) - U^E_m(p))\Big] I_m\tau \\
&\quad + \sum_{m=1}^{M}\mathbb{E}\Big[(L^E_m(p) - r(\lambda_p, \mu_p, \nu_p, \eta_p, p)\Big] I_m\tau.
\end{aligned} \tag{B.41}$$

We first analyze the middle term of (B.41). Let $\mathcal{M}_p$ denote the batches choosing price p and let $T_p = \sum_{m\in\mathcal{M}_p} I_m\tau$, then we have a same result as that in (4.17):

$$\sum_{m=1}^{M}\mathbb{E}\Big[(U^E_m(p) - U^E_m(p))\Big] I_m\tau \leq O\left(\sqrt{PT\log(T)}\right). \tag{B.42a}$$

Then we also have the same results for the first and the third terms of (B.41), and thus we omit the details. The key is the concentration bound we derived in Proposition 4.6.2 for this extended

system.

Putting the three terms together, we can derive that

$$\begin{aligned}\overline{\text{Regret}}^{\text{IND}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) =& \sum_{m=1}^{M} \mathbb{E}\Big[\Delta(p_m) I_m \tau\Big] + \sum_{m=1}^{M} O(\tau) \\ =& O\left(\sqrt{PT\log(T)}\right) + O\left(\frac{1}{T^3}\right) + O\big((\log(T))^2(\log\log(T))^2\big) \\ =& O\left(\sqrt{PT\log(T)}\right).\end{aligned} \tag{B.43}$$

Total Regret of the extended BUCB algorithm. By adding up the relaxed regret during the Warm-up and Learning Phases, we have the regret of the extend BUCB algorithm as

$$\overline{\text{Regret}}^{\text{IND}}(\pi_{\text{BUCB}}, T) = \overline{\text{Regret}}^{\text{IND}}_{\text{Warm-up}}(\pi_{\text{BUCB}}, T) + \overline{\text{Regret}}^{\text{IND}}_{\text{Learning}}(\pi_{\text{BUCB}}, T) \tag{B.44a}$$

$$= O(P\log(T)) + O\left(\sqrt{PT\log(T)}\right). \tag{B.44b}$$

Q.E.D.

Proof of Theorem 4.6.2. The analysis here is similar to the proof of Theorem 4.4.2.

Bayesian regret in the Warm-up Phase. Similarly, we can bound the Bayesian regret

$$\overline{\text{BR}}^{\text{IND}}_{\text{Warm-up}}(\pi_{\text{BTS}}, T) = O(P\log(T)). \tag{B.45}$$

Bayesian regret in the Learning Phase. The Bayesian regret in batch m is:

$$\overline{\text{BR}}^{\text{IND}}_{m} = \mathop{\mathbb{E}}_{(\lambda,\mu,\nu,\eta)\sim\text{prior}} \big[\mathbb{E}[J^{\text{IND}}_m - J_m(p_m)|\mathcal{F}_m]\big] \tag{B.46a}$$

$$= \mathop{\mathbb{E}}_{(\lambda,\mu,\nu,\eta)\sim\text{prior}} \big[\mathbb{E}[I_m\tau U^E_m(p_m) - J_m(p_m) + J^{\text{IND}}_m - I_m\tau U^E_m(\tilde{p})|\mathcal{F}_m]\big] \tag{B.46b}$$

$$= \mathop{\mathbb{E}}_{(\lambda,\mu,\nu,\eta)\sim\text{prior}} \big[\mathbb{E}[I_m\tau U^E_m(p_m) - J_m(p_m)|\mathcal{F}_m]\big] + \mathop{\mathbb{E}}_{(\lambda,\mu,\nu,\eta)\sim\text{prior}} \big[\mathbb{E}[J^{\text{IND}}_m - I_m\tau U^E_m(\tilde{p})|\mathcal{F}_m]\big] \tag{B.46c}$$

$$= \mathbb{E}[I_m\tau U^E_m(p_m) - J_m(p_m)] + \mathbb{E}[J^{\text{IND}}_m - I_m\tau U^E_m(\tilde{p})] \tag{B.46d}$$

$$\leq \mathbb{E}\left[I_m\tau U^E_m(p_m) - I_m\tau\cdot r^E(\lambda_{p_m},\mu_{p_m},\nu_{p_m},\eta_{p_m},p_m) + O(\tau)\right] + \mathbb{E}\left[I_m\tau\cdot r^E(\lambda_{\tilde{p}},\mu_{\tilde{p}},\nu_{\tilde{p}},\eta_{\tilde{p}},\tilde{p}) - I_m\tau U^E_m(\tilde{p})\right] \tag{B.46e}$$

$$= I_m\tau\left(\mathbb{E}\left[U^E_m(p_m) - r^E(\lambda_{p_m},\mu_{p_m},\nu_{p_m},\eta_{p_m},p_m)\right] + \mathbb{E}\left[r^E(\lambda_{\tilde{p}},\mu_{\tilde{p}},\nu_{\tilde{p}},\eta_{\tilde{p}},\tilde{p}) - U^E_m(\tilde{p})\right]\right) + O(\tau). \tag{B.46f}$$

Following the two properties (4.52) and (4.53), which are derived from Proposition 4.6.2, we have

$$\begin{aligned}
&\mathbb{E}\left[r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U_m^E(\tilde{p}) - U_m^E(\tilde{p})\right] \\
\leq &\mathbb{E}\left[(r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U_m^E(\tilde{p}) - U_m^E(\tilde{p}))^+\right] \\
\leq &\sum_{p\in\mathcal{P}} \mathbb{E}\left[(r^E(\lambda_p, \mu_p, \nu_p, \eta_p, p) - U_m^E(p) - U_m^E(p))^+\right] \\
\leq &\frac{\gamma}{T},
\end{aligned} \tag{B.47}$$

and

$$\begin{aligned}
&\mathbb{E}\left[U_m^E(p_m) - r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m)\right] \\
=&\mathbb{E}\left[2\cdot\mathbf{Rad}_m^E(p_m) + L_m^E(p_m) - r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m)\right] \\
\leq&\mathbb{E}\left[2\cdot\mathbf{Rad}_m^E(p_m)\right] + \sum_{p\in\mathcal{P}} \mathbb{E}\left[L_m^E(p) - r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m)\right] \\
\leq&\mathbb{E}\left[(2\cdot\mathbf{Rad}_m^E(p_m)\right] + \sum_{p\in\mathcal{P}} \mathbb{E}\left[(L_m^E(p) - r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m))^+\right] \\
\leq&\mathbb{E}\left[2\cdot\mathbf{Rad}_m^E(p_m)\right] + \frac{\gamma}{T}.
\end{aligned} \tag{B.48}$$

Summing over all batches $m \in [M]$, the relaxed Bayesian regret by the end of Learning Phase

$$\overline{\mathrm{BR}}^{\mathrm{IND}}_{\mathrm{Learning}}(\pi_{\mathrm{BTS}}, T) = \sum_{m=1}^{M} \overline{\mathrm{BR}}^{\mathrm{IND}}_m \tag{B.49a}$$

$$= \sum_{m=1}^{M} I_m \tau \left(\mathbb{E}\left[U_m^E(p_m) - r^E(\lambda_{p_m}, \mu_{p_m}, \nu_{p_m}, \eta_{p_m}, p_m)\right] + \mathbb{E}\left[r^E(\lambda_{\tilde{p}}, \mu_{\tilde{p}}, \nu_{\tilde{p}}, \eta_{\tilde{p}}, \tilde{p}) - U_m^E(\tilde{p})\right]\right) + O(\tau) \tag{B.49b}$$

$$\leq \sum_{m=1}^{M} I_m \tau \left(2\frac{\gamma}{T} + 2\mathbb{E}\left[\mathbf{Rad}_m^E(p_m)\right]\right) + O(\tau\log(T)) \tag{B.49c}$$

$$\leq 2\sum_{m=1}^{M} I_m \tau \frac{\gamma}{T} + 2\sum_{p\in\mathcal{P}}\sum_{m\in\mathcal{M}_p} I_m \tau \mathbb{E}\left[\mathbf{Rad}_m^E(p_m)\right] + O(\log(T)^2(\log\log(T))^2) \tag{B.49d}$$

$$= 2\gamma + \sum_{p\in\mathcal{P}} O(\sqrt{T_p\log(T)}) + O(\log(T)^2(\log\log(T))^2) \tag{B.49e}$$

$$= O(\sqrt{PT\log(T)}) + O(\log(T)^2(\log\log(T))^2) \tag{B.49f}$$

$$= O(\sqrt{PT\log(T)}), \tag{B.49g}$$

where (B.49e) is by Lemma B.4.1 and (B.49f) is by Jensen's inequality.

Regret of extended BTS algorithm. By adding up the relaxed regret during the Warm-up and

Learning Phases, we have the regret of BTS algorithm as

$$\overline{\mathrm{BR}}^{\mathrm{IND}}(\pi_{\mathrm{BTS}}, T) = \overline{\mathrm{BR}}^{\mathrm{IND}}_{\text{Warm-up}}(\pi_{\mathrm{BTS}}, T) + \overline{\mathrm{BR}}^{\mathrm{IND}}_{\text{Learning}}(\pi_{\mathrm{BTS}}, T) \tag{B.50a}$$

$$= O(P\log(T)) + O\left(\sqrt{PT\log(T)}\right). \tag{B.50b}$$

Q.E.D.

APPENDIX C

Appendix For Chapter 5

C.1 Summary of Major Notation

Table C.1: Summary of Major Notation for Problem Formulation

$\mathcal{I}$	the set of reusable resource (server) types
$\mathcal{J}$	the set of product (service) types
c_i	the capacity of server type $i \in \mathcal{I}$
$\mu_i(\mathbf{x})$	the service rate of server type i under the price $\mathbf{x}$
$\mathbf{x}$	the price vector $\mathbf{x} = \{x_j : j \in \mathcal{J}\}$
$\mathcal{X}$	the set of candidate prices
T	the number of periods in consideration
$\alpha(\mathbf{x})$	the overall demand rate of the system under the price $\mathbf{x}$
$\alpha_j(\mathbf{x})$	the demand rate of product j under the price $\mathbf{x}$
$\vec{\alpha}_{\mathbf{x}}$	the vector to demand of products under the price $\mathbf{x}$, $\vec{\alpha}_{\mathbf{x}} \in \mathbb{R}^{\lvert\mathcal{J}\rvert}$
$\lambda_i(\mathbf{x})$	the demand rate of server i under the price $\mathbf{x}$
A	the consuming matrix, $A_{ji} \in \mathbb{R}^{\lvert\mathcal{J}\rvert\times\lvert\mathcal{I}\rvert}$, where element $a_{ji} = 1$ indicates that service type j requires to be processed on server type i
$\mathcal{J}_{\text{on}}(\mathbf{x})$	the turn-on product set under price $\mathbf{x}$, containing products with non-zero demand rates
$\mathcal{J}_{\text{off}}(\mathbf{x})$	the turn-off product set under price $\mathbf{x}$, containing products with zero demand rates
$\mathcal{P}$	the server routing matrix where $p_{ii'}$ denotes the demand portion that consumes resource i' straight after consuming resource i, $\sum_{i' \in \mathcal{I}\cup\{t\}} p_{ii'} = 1$.
J_t^π	the expected revenue under pricing policy π during period t, $\forall t = 1, \ldots, T$,
J^π	the cumulative expected revenue over periods $\{1, \ldots, T\}$, $J^\pi = \sum_{t=1}^{T} J_t^\pi$
$\tilde{\mathbf{x}}$	the (static) optimal state-independent price, $\tilde{\mathbf{x}} = \text{argmax}_{\mathbf{x}\in\mathcal{X}} \vec{\alpha}_{\mathbf{x}}^T \mathbf{x}$

BIBLIOGRAPHY

Abate J, Whitt W (1987) Transient behavior of the m/m/l queue: Starting at the origin. *Queueing systems* 2(1):41–65.

Abate J, Whitt W (1988) Transient behavior of the m/m/1 queue via laplace transforms. *Advances in Applied Probability* 20(1):145–178.

Abbasi-Yadkori Y, Pál D, Szepesvári C (2011) Improved algorithms for linear stochastic bandits. *Advances in neural information processing systems* 24:2312–2320.

Agrawal S, Devanur N (2016) Linear contextual bandits with knapsacks. *Advances in Neural Information Processing Systems* 29.

Agrawal S, Goyal N (2013) Further optimal regret bounds for Thompson sampling. Carvalho CM, Ravikumar P, eds., *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, volume 31 of *Proceedings of Machine Learning Research*, 99–107 (Scottsdale, Arizona, USA: PMLR).

Ahuja RK, Magnanti TL, Orlin JB (1993) *Network Flows: Theory, Algorithms, and Applications.* (Prentice Hall, Upper Saddle River, New Jersey).

Akinc U, Khumawala BM (1977) An efficient branch and bound algorithm for the capacitated warehouse location problem. *Management Science* 23(6):585–594.

Al-Seedy RO, El-Sherbiny A, El-Shehawy S, Ammar S (2009) Transient solution of the M/M/c queue with balking and reneging. *Computers & Mathematics with Applications* 57(8):1280–1285.

Ammar SI, Helan MM, Al Amri FT (2013) The busy period of an m/m/1 queue with balking and reneging. *Applied mathematical modelling* 37(22):9223–9229.

Antonio N, de Almeida A, Nunes L (2019) Big data in hotel revenue management: Exploring cancellation drivers to gain insights into booking cancellation behavior. *Cornell Hospitality Quarterly* 60(4):298–319.

Araman VF, Caldentey R (2009) Dynamic pricing for nonperishable products with demand learning. *Operations Research* 57(5):1169–1188.

Auer P, Cesa-Bianchi N, Fischer P (2002a) Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47(2):235–256.

Auer P, Cesa-Bianchi N, Fischer P (2002b) Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47(2–3):235–256.

Auer P, Ortner R (2010) Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica* 61(1-2):55–65.

Badanidiyuru A, Kleinberg R, Slivkins A (2013) Bandits with knapsacks. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 207–216 (IEEE).

Bailey NT (1954) A continuous time treatment of a simple queue using generating functions. *Journal of the Royal Statistical Society: Series B (Methodological)* 16(2):288–291.

Baltean-Lugojan R, Bonami P, Misener R, Tramontani A (2019) Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks .

Banerjee S, Johari R, Riquelme C (2015) Pricing in ride-sharing platforms: A queueing-theoretic approach. *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, 639, EC '15 (New York, NY, USA: Association for Computing Machinery).

Beasley JE (1988) An algorithm for solving large capacitated warehouse location problems. *European Journal of Operational Research* 33(3):314–325.

Bell MG (2006) Mixed Route Strategies for the Risk-averse Shipment of Hazardous Materials. *Networks and Spatial Economics* 6(3-4):253–265.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4(1):238–252.

Besbes O, Elmachtoub AN, Sun Y (2020) Static pricing: Universal guarantees for reusable resources. Technical report, Columbia University, New York, NY. Forthcoming in *Operations Research*.

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.

Bimpikis K, Candogan O, Saban D (2019) Spatial pricing in ride-sharing networks. *Operations Research* 67(3):744–769.

Binato S, Pereira MVF, Granville S (2001) A new Benders decomposition approach to solve power transmission network design problems. *IEEE Transactions on Power Systems* 16(2):235–240.

Birge JR, Louveaux F (2011) *Introduction to Stochastic Programming* (Springer).

Borgs C, Candogan O, Chayes J, Lobel I, Nazerzadeh H (2014) Optimal multiperiod pricing with service guarantees. *Management Science* 60(7):1792–1811.

Boucheron S, Lugosi G, Massart P (2013) *Concentration inequalities: A nonasymptotic theory of independence* (Oxford university press).

Broder J, Rusmevichientong P (2012) Dynamic pricing under a general parametric choice model. *Operations Research* 60(4):965–980.

Bubeck S, Cesa-Bianchi N, Lugosi G (2013) Bandits with heavy tail. *IEEE Transactions on Information Theory* 59(11):7711–7717.

Burke PJ (1956) The output of a queuing system. *Operations research* 4(6):699–704.

Cai X, McKinney DC, Lasdon LS, Watkins Jr DW (2001) Solving large nonconvex water resources management models using generalized Benders decomposition. *Operations Research* 49(2):235–245.

Campigotto P, Rudloff C, Leodolter M, Bauer D (2016) Personalized and situation-aware multimodal route recommendations: the favour algorithm. *IEEE Transactions on Intelligent Transportation Systems* 18(1):92–102.

Cesa-Bianchi N, Dekel O, Shamir O (2013) Online learning with switching costs and other adaptive adversaries. *Advances in Neural Information Processing Systems*, 1160–1168.

Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2(3):1–27.

Chapelle O, Li L (2011) An empirical evaluation of Thompson sampling. *Proceedings of the 24th International Conference on Neural Information Processing Systems*, 2249–2257, NIPS'11 (Red Hook, NY, USA: Curran Associates Inc.).

Chen B, Chao X (2019) Parametric demand learning with limited price explorations in a backlog stochastic inventory system. *IISE Transactions* 51(6):605–613.

Chen B, Chao X, Wang Y (2020a) Data-based dynamic pricing and inventory control with censored demand and limited price changes. *Operations Research* 68(5):1445–1456.

Chen B, Shi C (2019a) Tailored base-surge policies in dual-sourcing inventory systems with demand learning. *Available at SSRN 3456834* .

Chen Q, Jasin S, Duenyas I (2019) 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, Liu Y, Hong G (2020b) An online learning approach to dynamic pricing and capacity sizing in service systems. Technical report, Chinese University of Hong Kong, Shenzhen, China.

Chen Y, Levi R, Shi C (2017) Revenue management of reusable resources with advanced reservations. *Production and Operations Management* 26(5):836–859.

Chen Y, Shi C (2017) Optimal pricing policy for service systems with reusable resources and forward-looking customers. Technical report, University of Michigan, Ann Arbor, MI.

Chen Y, Shi C (2019b) Network revenue management with online inverse batch gradient descent method. *Available at SSRN* .

Cheung WC, Simchi-Levi D, Wang H (2017) Dynamic pricing and demand learning with limited price experimentation. *Operations Research* 65(6):1722–1731.

Christofides N, Beasley JE (1983) Extensions to a lagrangean relaxation approach for the capacitated warehouse location problem. *European Journal of Operational Research* 12(1):19–28.

Cordeau JF, Stojković G, Soumis F, Desrosiers J (2001) Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science* 35(4):375–388.

Cortes C, Vapnik V (1995) Support-vector networks. *Machine Learning* 20(3):273–297.

Costa AM (2005) A survey on Benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research* 32(6):1429–1450.

Crainic TG, Frangioni A, Gendron B (2001) Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics* 112(1-3):73–99.

Crainic TG, Fu X, Gendreau M, Rei W, Wallace SW (2011) Progressive hedging-based metaheuristics for stochastic network design. *Networks* 58(2):114–124.

Crainic TG, Hewitt M, Rei W (2014) *Partial decomposition strategies for two-stage stochastic integer programs* (CIRRELT).

Cui G, Luo J, Wang X (2018) Personalized travel route recommendation using collaborative filtering based on gps trajectories. *International journal of digital earth* 11(3):284–307.

Dai J, Yang B, Guo C, Ding Z (2015) Personalized route recommendation using big trajectory data. *2015 IEEE 31st International Conference on Data Engineering*, 543–554 (IEEE).

Dai JG, Gluzman M (2017) Queueing network controls via deep reinforcement learning. Technical report, Cornell University, Ithaca, NY.

Daley DJ, Servi L (1998) Idle and busy periods in stable m/m/k queues. *Journal of applied probability* 35(4):950–962.

Dashti H, Conejo AJ, Jiang R, Wang J (2016) Weekly two-stage robust generation scheduling for hydrothermal power systems. *IEEE Transactions on Power Systems* 31(6):4554–4564.

Delling D, Goldberg AV, Pajor T, Werneck RF (2011) Customizable route planning. *International Symposium on Experimental Algorithms*, 376–387 (Springer).

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.

Diaconis P, Ylvisaker D (1979) Conjugate priors for exponential families. *The Annals of statistics* 269–281.

Doan XV, Lei X, Shen S (2020) Pricing of reusable resources under ambiguous distributions of demand and service time with emerging applications. *European Journal of Operational Research* 282(1):235–251.

Dubey A, Pentland A (2019) Thompson sampling on symmetric α-stable bandits. *arXiv preprint arXiv:1907.03821* .

Dubey A, et al. (2020) Cooperative multi-agent bandits with heavy tails. *International Conference on Machine Learning*, 2730–2739 (PMLR).

Federgruen A, Zipkin P (1984) A combined vehicle routing and inventory allocation problem. *Operations Research* 32(5):1019–1037.

Ferreira KJ, Simchi-Levi D, Wang H (2018) Online network revenue management using Thompson sampling. *Operations Research* 66(6):1586–1602.

Feurer M, Klein A, Eggensperger K, Springenberg J, Blum M, Hutter F (2015) Efficient and robust automated machine learning. *Advances in neural information processing systems*, 2962–2970.

Filippi S, Cappe O, Garivier A, Szepesvári C (2010) Parametric bandits: The generalized linear case. *NIPS*, volume 23, 586–594.

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.

Gans N, Savin S (2007) Pricing and capacity rationing for rentals with uncertain durations. *Management Science* 53(3):390–407.

Gao Z, Han Y, Ren Z, Zhou Z (2019) Batched multi-armed bandits problem. *arXiv preprint arXiv:1904.01763* .

Garivier A, Moulines E (2011) On upper-confidence bound policies for switching bandit problems. *International Conference on Algorithmic Learning Theory*, 174–188 (Springer).

Gavriel C, Hanasusanto G, Kuhn D (2012) Risk-averse Shortest Path Problems. *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 2533–2538 (IEEE).

Geisberger R, Sanders P, Schultes D, Delling D (2008) Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *International Workshop on Experimental and Efficient Algorithms*, 319–333 (Springer).

Gendron B, Scutellà MG, Garroppo RG, Nencioni G, Tavanti L (2016) A branch-and-Benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research* 255(1):151–162.

Geoffrion AM, Graves GW (1974) Multicommodity distribution system design by Benders decomposition. *Management Science* 20(5):822–844.

Hadar J, Russell WR (1969) Rules for ordering uncertain prospects. *The American economic review* 59(1):25–34.

Han Y, Zhou Z, Zhou Z, Blanchet J, Glynn PW, Ye Y (2020) Sequential batch learning in finite-action linear contextual bandits. *arXiv preprint arXiv:2004.06321* .

He F, Yin Y, Lawphongpanich S (2014a) Network Equilibrium Models with Battery Electric Vehicles. *Transportation Research Part B: Methodological* 67:306–319.

He H, Daume III H, Eisner JM (2014b) Learning to search in branch and bound algorithms. *Advances in Neural Information Processing Systems (NIPS)*, 3293–3301.

Holmberg K (1990) On the convergence of cross decomposition. *Mathematical Programming* 47(1-3):269–296.

Hooker JN (2007) Planning and scheduling by logic-based Benders decomposition. *Operations Research* 55(3):588–602.

Jackson JR (1957) Networks of waiting lines. *Operations research* 5(4):518–521.

Jackson JR (1963) Jobshop-like queueing systems. *Management science* 10(1):131–142.

Jia H, Fang J, Tan N, Liu X, Huo Z, Ma N, Wu G, Chai H, Qie X, Zhang B, Yin Y, Shen S (2020a) Context-aware route recommendation with weight learning through deep neural networks. *2020 American Control Conference (ACC)*, 4040–4045 (IEEE).

Jia H, Shen S (2021) Benders cut classification via support vector machines for solving two-stage stochastic programs. *INFORMS Journal on Optimization* 3(3):278–297.

Jia H, Shi C, Shen S (2020b) Online learning and pricing for service systems with reusable resources. *Available at SSRN 3755902* .

Jia H, Shi C, Shen S (2021) Multi-armed bandit with sub-exponential rewards. *Operations Research Letters* 49(5):728–733.

Kaewpuang R, Niyato D, Wang P, Hossain E (2013) A framework for cooperative resource management in mobile cloud computing. *IEEE Journal on Selected Areas in Communications* 31(12):2685–2700.

Kaufmann E, Korda N, Munos R (2012) Thompson sampling: An asymptotically optimal finite-time analysis. *International Conference on Algorithmic Learning Theory*, 199–213 (Springer).

Kelton WD, Law AM (1985) The transient behavior of the m/m/s queue, with implications for steady-state simulation. *Operations Research* 33(2):378–396.

Keskin NB, Zeevi A (2014) Dynamic pricing with an unknown demand model: Asymptotically optimal semi-myopic policies. *Operations research* 62(5):1142–1167.

Khalil E, Dai H, Zhang Y, Dilkina B, Song L (2017) Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, 6348–6358.

Khalil EB, Le Bodic P, Song L, Nemhauser GL, Dilkina BN (2016) Learning to branch in mixed integer programming. *Proceedings of the thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, 724–731.

Kleywegt AJ, Shapiro A, Homem-de Mello T (2002) The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization* 12(2):479–502.

Klibi W, Martel A (2012) Scenario-based supply chain network risk modeling. *European Journal of Operational Research* 223(3):644–658.

Klibi W, Martel A, Guitouni A (2010) The design of robust value-creating supply chain networks: a critical review. *European Journal of Operational Research* 203(2):283–293.

Korda N, Kaufmann E, Munos R (2013) Thompson sampling for 1-dimensional exponential family bandits. *arXiv preprint arXiv:1307.3400* .

Kruber M, Lübbecke ME, Parmentier A (2017) Learning when to use a decomposition. *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, 202–210 (Springer).

Lai TL (1987) Adaptive treatment allocation and the multi-armed bandit problem. *The Annals of Statistics* 1091–1114.

Lai TL, Robbins H (1985) Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6(1):4–22.

Laporte G (1992) The Vehicle Vouting Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research* 59(3):345–358.

Laporte G, Louveaux FV, Mercure H (1994) A priori optimization of the probabilistic traveling salesman problem. *Operations Research* 42(3):543–549.

Lattimore T, Szepesvári C (2020) *Bandit Algorithms* (Cambridge University Press, Cambridge, UK).

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *nature* 521(7553):436.

Ledermann W, Reuter GEH (1954) Spectral theory for the differential equations of simple birth and death processes. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences* 246(914):321–369.

Ledoux M (2001) *The Concentration of Measure Phenomenon*. Mathematical surveys and monographs (American Mathematical Society), ISBN 9780821837924, URL https://books.google.com/books?id=mCX_cWL6rqwC.

Lei Y, Jasin S (2020) Real-time dynamic pricing for revenue management with reusable resources, advance reservation, and deterministic service time requirements. *Operations Research* 68(3):676–685.

Lei YM, Jasin S, Sinha A (2014) Near-optimal bisection search for nonparametric dynamic pricing with inventory constraint. *Ross School of Business Paper* (1252).

Letchner J, Krumm J, Horvitz E (2006) Trip router with individualized preferences (trip): Incorporating personalization into route planning. *AAAI*, 1795–1800.

Levi R, Radovanović A (2010) Provably near-optimal LP-based policies for revenue management in systems with reusable resources. *Operations Research* 58(2):503–507.

Li L, Chu W, Langford J, Schapire RE (2010) A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web*, 661–670.

Liu B, Xie Q, Modiano E (2019) Reinforcement learning for optimal control of queueing systems. *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 663–670.

Liu Y, Li Y (2017) Pricing scheme design of ridesharing program in morning commute problem. *Transportation Research Part C: Emerging Technologies* 79:156–177.

Lou Y, Yin Y, Lawphongpanich S (2010) Robust Congestion Pricing under Boundedly Rational User Equilibrium. *Transportation Research Part B: Methodological* 44:15–28.

Maglaras C (2006) Revenue management for a multiclass single-server queue via a fluid model analysis. *Operations Research* 54(5):914–932.

Magnanti TL, Wong RT (1981) Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research* 29(3):464–484.

Massey WA (1984) An operator-analytic approach to the jackson network. *Journal of Applied Probability* 21(2):379–393.

Massey WA (1986) A family of bounds for the transient behavior of a jackson network. *Journal of applied probability* 23(2):543–549.

Massey WA (1987) Stochastic orderings for markov processes on partially ordered spaces. *Mathematics of operations research* 12(2):350–367.

Melkote S, Daskin MS (2001) Capacitated facility location/network design problems. *European journal of operational research* 129(3):481–495.

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* .

Minoux M (1986) *Mathematical Programming: Theory and Algorithms* (John Wiley & Sons).

Misra S, Roald L, Ng Y (2018) Learning for constrained optimization: Identifying optimal active constraint sets. *arXiv preprint arXiv:1802.09639* .

Mohamed AM, Lipsky L, Ammar R (2003) Transient model for jackson networks and its approximation. *International Conference On Principles Of Distributed Systems*, 198–209 (Springer).

Mohamed AM, Lipsky L, Ammar R (2005) Modeling parallel and distributed systems with finite workloads. *Performance Evaluation* 60(1-4):303–325.

Morisaku T (1976) *Techniques for data-truncation in digital computer simulation.* (University of Southern California).

Naoum-Sawaya J, Elhedhli S (2013) An interior-point Benders based branch-and-cut algorithm for mixed integer programs. *Annals of Operations Research* 210(1):33–55.

Oliveira BB, Carravilla MA, Oliveira JF (2018) Integrating pricing and capacity decisions in car rental: A matheuristic approach. *Operations Research Perspectives* 5:334–356.

Omahen K, Marathe V (1978) Analysis and applications of the delay cycle for the m/m/c queueing system. *Journal of the ACM (JACM)* 25(2):283–303.

Orchard-Hays W, et al. (1968) *Advanced Linear-programming Computing Techniques* (McGraw-Hill).

Owen Z, Simchi-Levi D (2018) Price and assortment optimization for reusable resources. *Available at SSRN 3070625* .

Özer Ö, Ozer O, Phillips R (2012) *The Oxford Handbook of Pricing Management* (Oxford University Press).

Parthasarathy P, Sharafali M (1989) Transient solution to the many-server poisson queue: A simple approach. *Journal of applied probability* 26(3):584–594.

Perchet V, Rigollet P, Chassang S, Snowberg E (2016) Batched bandit problems. *The Annals of Statistics* 44(2):660–681.

Pomerleau DA (1991) Efficient training of artificial neural networks for autonomous navigation. *Neural Computation* 3(1):88–97.

Püschel T, Schryen G, Hristova D, Neumann D (2015) Revenue management for cloud computing providers: Decision models for service admission control under non-probabilistic uncertainty. *European Journal of Operational Research* 244(2):637–647.

Rahmaniani R, Crainic TG, Gendreau M, Rei W (2017) The Benders decomposition algorithm: A literature review. *European Journal of Operational Research* 259(3):801–817.

Raidl GR (2015) Decomposition based hybrid metaheuristics. *European Journal of Operational Research* 244(1):66–76.

Raiffa H, Schlaifer R (1961) Applied statistical decision theory .

Reddy S, Dragan AD, Levine S (2019) Sqil: Imitation learning via regularized behavioral cloning. *arXiv preprint arXiv:1905.11108* .

Ross S, Gordon G, Bagnell D (2011) A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 627–635.

Rusmevichientong P, Sumida M, Topaloglu H (2020) Dynamic assortment optimization for reusable products with random usage durations. *Management Science* 66(7):2820–2844.

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.

Russo DJ, Roy BV, Kazerouni A, Osband I, Wen Z (2018) A tutorial on Thompson sampling. *Foundations and Trends in Machine Learning* 11(1):1–96.

Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T (2016) One-shot learning with memory-augmented neural networks. *arXiv preprint arXiv:1605.06065* .

Saravanan B, Das S, Sikri S, Kothari D (2013) A solution to the unit commitment problem—a review. *Frontiers in Energy* 7(2):223–236.

Savin SV, Cohen MA, Gans N, Katalan Z (2005) Capacity management in rental businesses with two customer bases. *Operations Research* 53(4):617–631.

Schölkopf B, Smola AJ, Bach F, et al. (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (MIT Press).

Seber GA, Lee AJ (2012) *Linear regression analysis*, volume 329 (John Wiley & Sons).

Seth S, Yalonetzky G (2014) Stochastic dominance with parametric distributions .

Shah D, Xie Q, Xu Z (2020) Stable reinforcement learning with unbounded state space. Bayen AM, Jadbabaie A, Pappas G, Parrilo PA, Recht B, Tomlin C, Zeilinger M, eds., *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, volume 120 of *Proceedings of Machine Learning Research*, 581–581 (The Cloud: PMLR).

Shalev-Shwartz S, et al. (2011) Online learning and online convex optimization. *Foundations and trends in Machine Learning* 4(2):107–194.

Shen S, You M, Ma Y (2017) Single-commodity Stochastic Network Design under Demand and topological Uncertainties with Insufficient Data. *Naval Research Logistics (NRL)* 64(2):154–173.

Simchi-Levi D, Xu Y (2019) Phase transitions and cyclic phenomena in bandits with switching constraints. *Available at SSRN 3380783* .

Slivkins A, et al. (2019) Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12(1-2):1–286.

Smola AJ (2004) *A Tutorial on Support Vector Regression* (Kluwer Academic Publishers).

Song Y, Luedtke JR, Küçükyavuz S (2014) Chance-constrained binary packing problems. *INFORMS Journal on Computing* 26(4):735–747.

Song Y, Shen S (2016) Risk-averse Shortest Path Interdiction. *INFORMS Journal on Computing* 28(3):527–539.

Talluri KT, Van Ryzin GJ (2006) *The Theory and Practice of Revenue Management*, volume 68 (Springer Science & Business Media).

Tang Y, Agrawal S, Faenza Y (2019) Reinforcement learning for integer programming: Learning to cut. *arXiv preprint arXiv:1906.04859* .

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.

Thompson WR (1935) On the theory of apportionment. *American Journal of Mathematics* 57(2):450–456.

Vakili S, Liu K, Zhao Q (2013) Deterministic sequencing of exploration and exploitation for multi-armed bandit problems. *IEEE Journal of Selected Topics in Signal Processing* 7(5):759–767.

Vapnik V (1998) *Statistical Learning Theory. 1998*, volume 3 (Wiley, New York).

Vapnik V (2013) *The Nature of Statistical Learning Theory* (Springer).

Vapnik VN (1999) An overview of statistical learning theory. *IEEE Transactions on Neural Networks* 10(5):988–999.

Wang H, Wang H, Yi F, Wen H, Li G, Sun L (2018) Context-aware personalized path inference from large-scale gps snippets. *Expert Systems with Applications* 91:78–88.

Wang K, Li N, Jiang Z (2010) Queueing system with impatient customers: A review. *Proceedings of 2010 IEEE International Conference on Service Operations and Logistics, and Informatics*, 82–87 (IEEE).

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.

Ward AR (2012) Asymptotic analysis of queueing systems with reneging: A survey of results for FIFO, single class models. *Surveys in Operations Research and Management Science* 17(1):1–14.

Ward AR, Glynn PW (2003) A diffusion approximation for a Markovian queue with reneging. *Queueing Systems* 43(1-2):103–128.

Ward AR, Glynn PW (2005) A diffusion approximation for a GI/GI/1 queue with balking or reneging. *Queueing Systems* 50(4):371–400.

Williamson EL (1992) *Airline network seat inventory control: Methodologies and revenue impacts*. Ph.D. thesis, Massachusetts Institute of Technology.

Wolfe P (1970) Convergence theory in nonlinear programming. *Integer and Nonlinear Programming* 1–36.

Xia Y, Qin T, Ma W, Yu N, Liu TY (2016) Budgeted multi-armed bandits with multiple plays. *IJCAI*, 2210–2216.

Xu H, Li B (2013) Dynamic cloud pricing for revenue maximization. *IEEE Transactions on Cloud Computing* 1(2):158–171.

Xu H, Lou Y, Yin Y, Zhou J (2011) A Prospect-based User Equilibrium Model with Endogenous Reference Points. *Transportation Research Part B: Methodological* 45:311–328.

Yan C, Zhu H, Korolko N, Woodard D (2020) Dynamic pricing and matching in ride-hailing platforms. *Naval Research Logistics (NRL)* 67(8):705–724.

Yang B, Guo C, Ma Y, Jensen CS (2015) Toward personalized, context-aware routing. *The VLDB Journal—The International Journal on Very Large Data Bases* 24(2):297–318.

Zakeri G, Philpott AB, Ryan DM (2000) Inexact cuts in Benders decomposition. *SIAM Journal on Optimization* 10(3):643–657.

Zangui M, Yin Y, Lawphongpanich S, Chen S (2013) Differentiated Congestion Pricing of Urban Transportation Networks with Vehicle-tracking Technologies. *Transportation Research Part C: Emerging Technologies* 36:434–548.

Zhang Y, Yue D, Yue W (2005) Analysis of an M/M/1/N queue with balking, reneging and server vacations. *Proceedings of the 5th International Symposium on OR and its Applications*, 37–47.

Zhou Z, Xu R, Blanchet J (2019) Learning in generalized linear contextual bandits with stochastic delays. *Advances in Neural Information Processing Systems* 32:5197–5208.