# Essays on Dynamic Stochastic Matching, Customer Choice Modeling and Assortment Optimization in Online Marketplaces

by

**Fan You**

M.S., The Pennsylvania State University, 2015

B.S., University of Electronic Science and Technology of China, 2013

A dissertation submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy, Leeds School of Business

2022

You, Fan (Ph.D., Operations Management)

Essays on Dynamic Stochastic Matching, Customer Choice Modeling and Assortment Optimization in Online Marketplaces

Thesis directed by Associate Professor Thomas Vossen

**Abstract.** This dissertation comprises three essays on dynamic stochastic matching, customer choice modeling and assortment optimization. One common theme of the essays is developing efficient approximate solution strategies for challenging and practically relevant operational problems faced by online marketplaces. The first essay (Chapter 2) proposes a modeling framework for dynamic stochastic matching problems with a scalable and efficient solution strategy based on approximate dynamic programming. The second essay (Chapter 3) focuses on an important and unique feature of virtual item stores in online centralized marketplaces, which is that the seller has perfect information of its customers' complete purchase history and inventory; we study customers' choice behaviors under such considerations. The third essay (Chapter 4) analyzes virtual stores of online multiplayer video games and proposes to study the item recommendation problem through the lens of discrete choice modeling and assortment optimization; approximation algorithms as well as performance bounds are developed.

## Dedication

To my father Dunqiu You, and my mother Xiangqun Wang

献给我的父亲尤敦球，和我的母亲王向群

# Acknowledgements

First and foremost, I would like to express my deepest appreciation to my advisor, Dr. Thomas Vossen. Without his constant support and guidance throughout my entire doctoral studies, this dissertation would not have been possible. His curiosity, passion, creativity and rigor has motivated me throughout the last six years. I would like to also thank other members of my dissertation committee, Dr. Dan Zhang, Dr. Rui Zhang, Dr. Manuel Laguna, Dr. Huanan Zhang, and Dr. Spencer Stirling, for their helpful comments and suggestions. In particular, I am grateful to Dr. Stirling for the amazing year that I spent at Activision Blizzard. I am also grateful to other faculty members of the OM group at Leeds, and the supporting staff at Leeds. A special thanks goes to the PhD program manager Stefanie Ungphakorn, who is always there to assist myself and fellow PhD students in need.

I am fortunate to have been around wonderful peers and friends at CU. Although our journey was disrupted by the global pandemic, I will always cherish the conversations that we had in the office, and the powder days we had on the slopes.

I am grateful to students of the courses that I instructed, who taught me to be more precise when conveying information, and to always put the audience's interest first.

My sincerest gratitude goes to Luqi Yao, for the sacrifices she has to make for a long-distance relationship, and the financial and emotional support that she provided. I would like to thank Spirit Airlines and Frontier Airlines for the affordable flights, without which I would never have survived the program. Finally, I am forever indebted to my parents for their unconditional love and support.

# Contents

# Tables

# Figures

# Chapter 1

# Introduction

Online marketplaces are a type of e-commerce platforms where product or service information is provided by multiple parties. These online platforms provide a variety of technology-enabled tools that enhance market transparency and efficiency. Since the early 2000s, online marketplaces have been growing rapidly and transformed the modern society profoundly. In this dissertation, we investigate a few aspects of the operational challenges of online marketplaces, formulate the problems mathematically in a rigorous manner, and prescribe algorithmically efficient and effective solutions. In this introductory chapter, we first briefly review the development of online marketplaces of various types, and their key operational levers which are the main motivations of this work. We then discuss the specific problems that we address in this dissertation, while detailed literature reviews can be found in the respective chapters.

## 1.1. Online Marketplaces: Operational Challenges and Opportunities

In a traditional sense, online marketplaces are digital platforms that act as intermediaries by connecting buyers and sellers. Examples of prevalent online marketplaces for retailing consumer goods and services are Amazon, Taobao and eBay. Differentiated by participants, consumer retail marketplaces can be categorized into business-to-consumer (B2C) and consumer-to-consumer (C2C) marketplaces. Examples of B2C marketplaces include Amazon, Newegg, T-Mall and JD.com, where a centralized market operator owns and sells all the products on the platform. On the other hands, in C2C marketplaces, e.g., Etsy, Facebook Marketplace and Ebay, anyone can sign up and start to sell or buy products, and the marketplace operator offers logistics, online payments or other related services. Market operators of B2C marketplaces primarily face the challenge of understanding customer choice patterns, and then offer personalized pricing or assortments/recommendations to achieve improved market efficiency, customer satisfaction and revenue. In C2C marketplaces, the operator instead aims to match buyers and sellers that are more likely to result in transactions.

In addition to consumer retailing, the emergence of the so-called *gig economy* has lead to successful companies such as Uber and Airbnb, among many others, that have revolutionized the way supply and demand is connected. Following the digital transformation due to the development of information and communication technologies such as the Internet and the popularization of smartphones, on-demand platforms based on digital technology have created new jobs and employment forms, generating billions of dollars of revenue while growing rapidly. Gig economy companies span a variety business sectors, including accommodation (Airbnb, Vrbo), delivery (GoPuff, Shipt), grocery (Instacart), food delivery (Uber Eats, DoorDash), home services (thumbtack), personal transportation (Uber, Lyft), trucking (Convoy) and freight forwarding (Flexport).

In these gig economy platforms, one central operational task is to match supply and demand, which takes into account potential outcomes of different matchings, and stochastic arrivals and departures of agents; the availability of data collected by these digital platforms offer the opportunity to design intelligent algorithms for the matching decisions, which can lead to substantially higher market efficiency compared to traditional ad-hoc policies.

Another example of online marketplaces of particular interest is kidney exchange, where kidney failure patients with willing but medically incompatible donors swap donors so that more patients receive compatible kidneys. Kidney exchange programs have been successfully implemented across the globe, resulting in improved quality of transplants with fewer kidney failures, and reduced waiting time. Efficiently connecting patient-donor pairs requires specially designed algorithms, which can help save millions of dollars in health care cost and more importantly, patient lives.

## 1.2. Organization and Summary of Contributions

In this dissertation, we focus on three problems faced by online marketplaces, identify the potential benefits of formulation and solution of the problems, and develop specialized approaches that are practically relevant, easy to implement, and effective.

In Chapter 2, we study the stochastic dynamic matching problem faced by gig economy platforms as well as kidney exchange programs. Matching problems have been an important area of research in Operations Research, Computer Science and Economics in the last century with numerous influential results. In the digital era, the access to historic data and the need for real-time decisions ask for technical innovations that address the stochastic and uncertain nature of the problem. In particular, we propose to formulate the problem of connecting users of these platforms as discrete-time Markov decision processes, which take into account stochastic arrival and departures, potential failures, and heterogeneous reward structures. Such Markov decision processes are difficult to solve, and we propose an linear programming based approximate dynamic programming framework which provides high quality performance bounds and control policies. Our framework extends existing research in LP-based ADP, and we show its applicability with an extensive numerical study.

In Chapter 3, we study the customer choice modeling problem faced by centralized B2C platforms. Customer choice modeling is a fundamental task for B2C platforms upon which personalized pricing and assortment recommendation systems are built. One unique feature of the centralized B2C platforms that we consider is that the platform has complete assortment, purchase and inventory information of all the customers. Examples of these marketplaces include virtual item stores of online multiplayer video games, or centralized B2C marketplaces in the "metaverse". We explore ways in which the marketplace operator could leverage all the information to model customer choice behaviors, and propose a utility based model which we call the *inventory-aware multinomial logit model*. We conduct numerical experiments to evaluate the performance of our proposed approach against several benchmark approaches, and explore conditions under which our approach could be useful.

In Chapter 4, we study the assortment optimization problem faced by a class of online B2C marketplaces with a specific structure which involves a combination of non-personalized *Featured* section and personalized item recommendations. Examples of retailers with this structure include virtual item stores of online multiplayer video game and traditional B2C platforms such as JD.com. The structure of these marketplaces motivates us to model customer choices using a nested-logit model. We also address customer heterogeneity by assuming customer segmentation is readily available. Customer segmentation and choice model estimation can be achieved using techniques introduced in Chapter 3. The optimization problem of recommending assortments is the focus of this chapter, which is a computationally challenging problem. We develop a fully polynomial time approximation scheme, a fast MILP formulation, a fast heuristic and a performance upper bound, and establish the effectiveness of the approaches on synthetic instances.

# Chapter 2

# An Approximate Dynamic Programming Approach to Dynamic Stochastic Matching

**Abstract.** Dynamic stochastic matching problems arise in a variety of recent applications, ranging from ridesharing and online video games to kidney exchange. Based on the nature of the underlying matching problems and other key characteristics of the specific applications, a number of disparate approaches have been proposed for solving such problems. This motivates us to explore the common structure in such problems and propose a general framework that can provide a unified approach for a broad class of dynamic stochastic matching problems. Our framework can accommodate a variety of important problem characteristics and is flexible regarding the nature of agent compatibilities, possible restrictions on the matchings, and the potential for matching failures. We formulate this framework as a Markov decision process (MDP). Because the resulting MDPs are intractable, we resort to the linear programming-based approach to approximate dynamic programming and derive novel reformulations of the resulting approximate linear programs (ALPs). Even though these reformulations may be neither compact nor equivalent to the original ALPs, they can be solved efficiently and produce strong bounds and heuristic policies. Numerical experiments confirm that this approach shows strong performance and provides an attractive alternative for a broad range of applications that involve dynamic stochastic matching.

## 2.1. Introduction

Matching problems are among the most fundamental and extensively studied problems in combinatorial optimization, with numerous applications in economics, computer science, and operations management. Given some graph, matching problems aim to determine a set of edges where each node is incident to at most one other node in the set and certain criteria are optimized. Recently there has been considerable interest in **dynamic** matching problems, where nodes in the graph may appear and/or depart dynamically over time. To a large extent this interest is driven by novel applications, ranging from online advertisement allocation (Karp et al. 1990, Mehta et al. 2007, Feldman et al. 2009) and real-time ridesharing (Özkan and Ward 2020, Yan et al. 2019) to matchmaking for online multiplayer video games (Emek et al. 2016) and kidney exchange (Dickerson et al. 2012, 2018).

This has led to a number of disparate approaches for solving such problems, based on the nature of the underlying matching problems and other key characteristics of the specific applications. For example, Özkan and Ward (2020) propose a continuous linear program together with a myopic LP-based matching policy for their dynamic bipartite matching problem, while Emek et al. (2016) propose an on-line algorithm for a dynamic general matching problem. Dickerson et al. (2012) use an iterated local search method to approximate the future value of matches in a setting where additional constraints are imposed on the matches that are allowed. In this work, however, we aim to explore the commonality between these problems and propose a general framework that can provide a unified approach for a broad class of dynamic matching problems.

Specifically, we consider a general dynamic matching problem where agents arrive gradually over a finite time horizon. Compatibility between agents is characterized by a general compatibility graph, where each node represents an agent type. The decision maker observes the agents present in the system, and decides whether and how to match them given the compatibility graph. Successfully matched agents obtain a reward and leave the system, while unmatched agents either persist or depart. The decision maker's objective is to maximize the expected total reward over the time horizon. In this setting, the decision maker has to determine both *which* agents to match and *when* to match agents. Thus, the decision maker has to balance immediate matchings with the potential increase in reward from later matchings, while accounting for waiting costs and potential agent departures. Our setup can also accommodate additional restrictions on the feasible matchings and account for potential match failures, which may occur with a certain probability after the decision maker has committed to a matching in each time period. When a proposed match fails, affected agents are returned to the system. Such failures are common in kidney exchanges, see Dickerson et al. (2018). For example, in the United Network for Organ Sharing (UNOS) 93% of matches fail, while most matches fail at other kidney exchanges as well, e.g., Bray et al. (2015). Similar failures may occur in other matching applications, such as online dating, ridesharing and matchmaking for multiplayer video games.

The stochastic nature of this general failure-aware dynamic matching problem is naturally accommodated by formulating the problem as a Markov decision process (MDP). However, the resulting MDPs suffer from the three "curses of dimensionality" (Powell 2011, p. 3–5) in that the state, action, and outcome space of the MDP all grow exponentially with the size of the underlying graph. To achieve tractability, we therefore resort to approximate dynamic programming (ADP), see Powell (2011) and Bertsekas (2008). In particular, we use the linear programming-based approach to approximate dynamic programming, which was introduced in Schweitzer and Seidmann (1985) and further explored in De Farias and Van Roy (2003, 2004). The main idea behind this approach is to approximate the value function with a collection of pre-selected weighted basis functions, and determine these weights by solving an approximate linear program (ALP). This approach has been used successfully in a number of settings, such as network revenue management (Farias and Van Roy 2007, Zhang and Adelman 2009, Vossen and Zhang 2015) and health care management (Patrick et al. 2008, Samiedaluie et al. 2017).

The linear programming-based approach to approximate dynamic programming leads to large-scale linear programming problems that are difficult to solve even with current commercial solvers. A number of different approaches have been proposed to address this computational challenge, ranging from column generation (Adelman 2007, Zhang and Adelman 2009) and constraint sampling (De Farias and Van Roy 2004) to constraint violation learning (Lin et al. 2020). We instead take a different approach and extend the work in Tong and Topaloglu (2013) and Vossen and Zhang (2015), who derive compact equivalent reformulations of the resulting ALPs for network revenue management problems and find that directly solving these reformulations can be orders of magnitude faster than other approaches.

However, the system dynamics and action spaces are substantially more complicated when considering dynamic stochastic matching problems. As a result, the reformulations we derive may be neither *compact* nor *equivalent*. For general non-bipartite compatibility graphs, our reformulation transforms an ALP with an exponential number of variables to a LP with an exponential number of constraints. Nevertheless, we show that these reformulations can be solved significantly faster than other approaches. When additional constraints are imposed in the feasible matchings, our reformulation relaxes the original ALP. Still, these reformulations provide strong performance bounds and high-quality heuristic control policies. Overall therefore, we establish a general algorithmic framework that encompasses and extends a broad class of dynamic matching problems. To our knowledge, our model is the first to propose ADP approaches for dynamic matching problems with general non-bipartite compatibility graphs, and our numerical results show that taking advantage of distributional information can yield strong performance in this setting. Moreover, our results demonstrate effectiveness of ADP approaches for dynamic kidney exchange problems, where our ALP reformulation provides tight performance bounds.

To summarize, our contributions are as follows:

1. We propose a unified framework for a variety of dynamic stochastic matching applications that accounts for stochastic arrivals and departures of agents, arbitrary compatibility relations between agents by allowing general (non-bipartite) compatibility graphs, agent waiting costs, and potential match failures. Our framework can also incorporate certain additional restrictions on the matchings that are allowed, for settings where matchings induce a cycle of exchanges and it is desirable to limit the length of such cycles.

2. We introduce a linear programming-based approximate dynamic programming approach to approximately solve dynamic matching problems. We propose reformulations of the original ALPs that can be solved efficiently and have an intuitive interpretation. We also show the promise of this approach even if the reformulations are not equivalent and relax the original ALPs. Overall therefore, we believe that our results extend the literature on Approximate Linear Programming.

3. We evaluate the performance of our approach with a numerical study that considers a number of different applications, ranging from kidney exchange and real-time ridesharing to matchmaking for online multiplayer video games. We show that our approach yields tight upper bounds and high quality control policies over this broad set of applications.

The remainder of the chapter is organized as follows. Section 2.2 surveys related literature on applications that involve dynamic matching. Section 2.3 introduces the formulation of our dynamic matching problem. In Section 2.2, we introduces the ALP framework, describe the key ideas behind our approach, and derive reformulations of the original ALPs. We also introduce heuristic control policies based on the solution to these ALPs. Section 2.5 describes our numerical experiments. In the appendix, Section A.1 contains all technical proofs. Section A.2 illustrates connections with two other related problems. Section A.3 provides additional implementation details. Section A.4 provides additional numerical experiments and details on the data sets for the applications we consider. Section 2.6 concludes.

## 2.2. Related Literature

The theory of matching has enjoyed extensive attention with significant results in the field of combinatorial optimization, economics and computer science. In more traditional static matching problems, the compatibility graph is known and fixed, and the objective is to find a matching that maximizes a certain reward. When the resulting graphs are bipartite, finding a maximum weight matching is also called the assignment problem and the well-known Hungarian algorithm can be used to solve this problem. For static matching problems on general graphs, Tutte's Matching Theorem appeared in the 1940s, and approaches based on augmenting paths were first suggested in the 1950s (Cook et al. 2009). An efficient algorithm was first introduced in the landmark paper of Edmonds (1965), who first provided a complete linear description of the $b$-matching polytope using the so-called *blossom inequalities.* Blossom inequalities can be identified using a separation procedure, and Padberg and Rao (1982) devise an efficient combinatorial separation algorithm based on the Gomory-Hu cut-tree (Gomory and Hu 1961). We use this result to construct equivalent ALP reformulations for a certain class of dynamic matching problems. However, the matching problems that we study are dynamic in nature and do not assume a known and fixed compatibility graph.

Dynamic matching problems are a class of problems where agents arrive and depart stochastically over time. When the compatibility graphs are bipartite, the problems are also known as *dynamic assignment* problems. Dynamic assignment problems have a long history, beginning with Derman et al. (1972) – see Su and Zenios (2005) for a more recent review. Spivey and Powell (2004) also consider an approximate dynamic programming approach for a class of dynamic assignment problems. However, their approach relies on iteratively updating the basis function weights in their approximations by solving a sequence of optimization problems, whereas our approach determines the basis function weights that provide the best bound (given the approximation) using a single LP formulation.

An emerging application of such dynamic bipartite matching problems arises in ridesharing platforms that aim to optimally match drivers and customers over time. Our model is most closely related to the models proposed by Özkan and Ward (2020) and Aouad and Saritac (2019), who formulate ridesharing systems as an open queueing network where driver and customer arrivals are independent of the matching decisions. There is support for such an assumption in the work of Zhong et al. (2019), who use data from the ride-sharing company Didi to show that a Markovian queueing model with exogenous customer and driver behaviors provides good estimates for system performance measures during rush hours in China. Özkan and Ward (2020) propose a myopic randomized allocation policy based on a continuous LP formulation and establish its asymptotic optimality. Aouad and Saritac (2019) propose allocation policies that rely on a LP formulation that is based on a fluid relaxation. Their LP formulation has a similar structure to our ALP reformulations when the compatibility graphs are bipartite, albeit for a continuous time setting where matchings occur over an infinite horizon; in contrast to Özkan and Ward (2020), their allocation policies allow the possibility of waiting for possible higher reward matches. Unlike these approaches, our formulation relies on a discrete time horizon that can more easily incorporate time varying arrival and departure rates.

We also consider dynamic matching problems with general non-bipartite compatibility graphs. Such problems arise in matchmaking systems for online player-versus-player (PvP) video games, which are immensely popular and routinely attract millions of players. Matchmaking systems provide a crucial service by connecting players for online game sessions: a good matchmaking system can increase player experience and engagement, which in turn improves retention (Butcher 2008). While certain video games require matchings that involve large groups or competing teams, an important class of video games (such as the *StarCraft* and the *Street Fighter* franchises) involves pairing two players for direct competition. Even in such cases, however, determining an optimal set of matchings presents a significant challenge. Because players join the game at different times, the quality of the matchings (based on factors such as server latency and skill balance) has to be balanced with the time players spend waiting for a game to begin. Véron et al. (2014) and Xu et al. (2019) analyze waiting times in video games, and establish that this tradeoff can have a significant impact on user experience. Typically, video game companies rely on *ad-hoc* rule-based heuristic policies that are myopic in nature. These can lead to inefficiencies when perfect opponents emerge shortly after a player has been paired, aptly described as "Haste Makes Waste" in Emek et al. (2016). Emek et al. (2016) refer to this problem as the *min-cost perfect matching with delays*, and derive worst-case performance guarantees based on a randomized online algorithm. While our overall setting is similar to the one considered by Emek et al. (2016), a key difference is that we account for and explicitly incorporate the arrival time distributions of the different players. This distributional information is often readily available based on the historical arrival patterns of players, and can be used to improve matchmaking decisions.

Another application of particular interest is kidney exchange. A kidney exchange occurs when two patients swap incompatible donors. Such exchanges can also take the form of longer cycles, where each patient receives a kidney from the donor of the next pair in the cycle. To ensure that donors follow through after their intended recipient successfully receives a kidney from another donor, cycle lengths are restricted by the number of medically feasible simultaneous transplants. The static version of the kidney exchange problem therefore involves a generalization of the general matching problem, often called the *barter exchange clearing problem* (Abraham et al. 2007), where exchange cycles are restricted to some maximum length $L$. Roth et al. (2005) show that significantly better solutions can be obtained by allowing cycles of length 3 instead of allowing 2-cycles only. Abraham et al. (2007) show that the matching problem for kidney exchange is NP-Complete when the cycle length limit is finite and larger than two, and present an IP formulation and branch-and-price algorithm for cycles up to length 3. Subsequently, Glorie et al. (2014) show that higher efficiency can be achieved by allowing longer chains, and propose a specialized branch-and-price algorithm to solve kidney exhange problems with longer cycles.

Awasthi and Sandholm (2009) first proposed a dynamic version of the matching problem for kidney exchange, where exchanges are conducted continuously and the exchange is optimized with respect to future arrivals and departures. They show that more matches can be achieved by considering the future, though the approach does not scale to larger instances. Dickerson et al. (2012) propose off-line machine learning approaches to learn the *potentials* of structural elements of the graph. These potentials can be viewed estimates of the value functions in a corresponding dynamic program. In our model, the value functions are approximated systematically by the weighted sum of a collection of basis functions. Our framework also addresses the issue that matches might fail *after* the algorithm is committed. We adopt procedures in Dickerson et al. (2018), who adjust the edge-reward based on such failures. The ability to provide upper bounds is a key benefit of ADP approaches in this setting. Whereas Dickerson et al. (2012) use perfect hindsight bounds and Dickerson et al. (2018) do not provide upper bounds on the performance of the policies they propose, our ALP formulation can provide strong upper bounds on performance.

## 2.3. Model Formulation and Preliminaries

We consider a system where agents arrive and depart over time. We assume that the time horizon is finite and discrete. A planner needs to decide which of the present agents to match at the start of each time period based on their compatibilities. A reward is collected for every successful match, and successfully matched agents leave the system. However, we also account for possible match failures; when matches fail, the corresponding agents return to the system in the next period. The objective is to maximize the total expected reward by selecting which matches to execute .

We formulate this problem as a finite-horizon discrete-time Markov decision process (MDP), and use $\mathcal{T} = \{1, ..., T\}$ to represent the planning horizon. We assume that each agent has a type $i \in \mathcal{V} = \{1, \ldots, V\}$, where $\mathcal{V}$ represents the set of all agent types. At the start of each period $t$, the planner observes the agents present in the system. Thus, the state space in our MDP can be expressed as

$$\mathcal{S} = \{\mathbf{s} \in \mathbb{Z}_+^V : s_i \leq S, \forall i \in \mathcal{V}\}.$$

We assume a finite state space by imposing a uniform upper bound $S$ on the state variables. This is without loss of generality, and avoids the technical complications that come with an infinite-dimensional MDP.

Given a state $\mathbf{s}_t$ at the start of each period, the planner has to decide which agents to match. To allow for a general and unified framework, we represent the compatibility between agents using a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Intuitively, we can interpret this by imagining that each agent has an item that it wishes to exchange. A directed arc $(i, j) \in \mathcal{E}$ then reflects that type $i$ agents are willing to exchange their items for items held by type $j$ agents. An elementary cycle in $\mathcal{G}$ (where each node in $\mathcal{G}$ is visited at most once) will therefore represent a feasible exchange of items, where each agent receives the item of its successor in the cycle. Under this setup, the planner determines which exchanges to execute by selecting a collection of cycles in $\mathcal{G}$.

To formally represent the action space in our MDP, we first define $\mathcal{C}$ to be the set of all elementary cycles in the compatibility graph $\mathcal{G}$. We further let $L \in \mathbb{Z}_+$ be the maximum cycle length and use $\mathcal{C}(L)$ to denote the set of elementary cycles with a maximum lengths of $L$. For any $c \in \mathcal{C}(L)$, we introduce decision variables $x_c$ to represent the number of times cycle $c$ is scheduled for exchange. For any agent type $i \in \mathcal{V}$, we introduce constraints to ensure that the number of type $i$ agents selected for exchange does not exceed the number of such agents present in the current state. As a result, the action space can be defined as

$$\mathcal{A}(\mathbf{s}) = \{\mathbf{x} \in \mathbb{Z}_+^{|\mathcal{C}(L)|} : \sum_{c \in \mathcal{C}(L): i \in \mathcal{V}(c)} x_c \leq s_i, \forall i \in \mathcal{V}\},$$

where $\mathcal{V}(c)$ represents the nodes in cycle $c$. To express state-action pairs, we define

$$\mathcal{X} = \{(\mathbf{s}, \mathbf{x}) : \mathbf{s} \in \mathcal{S}, \mathbf{x} \in \mathcal{A}(\mathbf{s})\}.$$

While this approach might be somewhat unusual, we emphasize that this framework can easily accommodate traditional matching problems with an undirected compatibility graph. Specifically, we can duplicate arcs in both directions for each edge in the undirected compatibility graphs, and impose the further restriction that each matching can only include cycles of length two. However, it can also be useful to consider settings that relax this restriction. In kidney exchange markets, for example, the maximum cycle length allowed is typically three; alternatively, timeshare exchange markets correspond to a setting where no restrictions are imposed on the cycle length (see Section 2.7 for additional details on this application).

Our framework incorporates the possibility that planned matches can fail, following the notion of *failure-aware* matching proposed in Dickerson et al. (2018). As in Dickerson et al. (2018) and Özkan and Ward (2020), we assume that each potential exchange $(i, j) \in \mathcal{E}$ has an independent success probability $\rho_{i,j}$. Because the exchanges in a cycle $c$ will fail whenever at least one of the exchanges in $c$ fails, we can define the failure probability for a cycle $c$ as $\mu_c = 1 - \prod_{(i,j)\in\mathcal{E}(c)} \rho_{i,j}$ where $\mathcal{E}(c)$ represents the arcs in $c$. The reward function consists of two components: waiting cost of agents in the system, and successful matching reward. Let $w_{t,i}$ be the waiting cost of each type $i$ agent at the beginning of each period, and $r_{t,c}$ is the reward of a type $c$ cycle if the exchanges in the cycle are successful, we can therefore associate an **expected** reward with each planned matching $\mathbf{x} \in \mathcal{A}(\mathbf{s})$ as follows:

$$r_t(\mathbf{s}, \mathbf{x}) = \sum_{c \in \mathcal{C}(L)} (1 - \mu_c) r_{t,c} x_c - \sum_{i \in \mathcal{V}} w_{t,i} s_i.$$

To represent the transition function we first use $\mathbf{y}_t$ to represent the number of successful exchange cycles given that the planned matches $\mathbf{x}_t$ when $\mathbf{s}_t$ agents are present at the start of each period, and define the conditional probability

$$P^f(\mathbf{y}_t | \mathbf{x}_t) = \prod_{c \in \mathcal{C}(L)} \binom{x_{t,c}}{y_{t,c}} \mu_c^{x_{t,c} - y_{t,c}} (1 - \mu_c)^{y_{t,c}}.$$

After successfully matched agents leave the system the number of remaining agents equals $\mathbf{s}'_t(\mathbf{s}_t, \mathbf{y}_t)$, which is defined as

$$s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t) = s_{t,i} - \sum_{c \in \mathcal{C}(L) : i \in \mathcal{C}(L)} y_{t,c}.$$

Finally, the planner also observes departures $\mathbf{d}_t = (d_{t,1}, \ldots, d_{t,V})$ of unmatched agents and arrivals $\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,V})$ of new agents. Thus, the number of agents $\mathbf{s}_{t+1}$ present at the start of the next period equals

$$s_{t+1,i} = s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t) - d_{t,i} + a_{t,i}.$$

We assume that the departures of each agent type are independent and follow a binomial distribution with probability $\nu_i$ for each $i \in \mathcal{V}$. It follows that the joint distribution of departures is

$$P^d(\mathbf{d}_t | \mathbf{s}_t, \mathbf{y}_t) = \prod_{i \in \mathcal{V}} p_i^d(d_{t,i} | s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t)),$$

where

$$p_i^d(d_{t,i} | s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t)) = \binom{s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t)}{d_{t,i}} \nu_i^{d_{t,i}} (1 - \nu_i)^{s'_{t,i}(\mathbf{s}_t, \mathbf{y}_t) - d_{t,i}}.$$

We also assume that the new arrivals of each agent type are independent, and follow a discrete distribution with bounded support. Thus, the joint distribution of arrivals is

$$P^a(\mathbf{a}_t) = \prod_{i \in \mathcal{V}} p_i^a(a_{t,i}),$$

where $p_i^a(a_{t,i})$ represents the probability that $a_{t,i}$ type $i$ customers arrive during period $t$. As a result, the overall transition probabilities $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{x}_t)$ can be expressed as

$$P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{x}_t) = \sum_{y_t: y_t \le x_t} \sum_{d_t: d_t \le \mathbf{s}'_t(s_t, y_t)} \sum_{a_t} P^f(\mathbf{y}_t|\mathbf{x}_t) P^d(\mathbf{d}_t|\mathbf{s}_t, \mathbf{y}_t) P^a(\mathbf{a}_t) \mathbb{1}\{\mathbf{s}_{t+1} = \mathbf{s}'_t(\mathbf{s}_t, \mathbf{y}_t) - \mathbf{d}_t + \mathbf{a}_t\}.$$

A feasible policy $\pi$ is a time-dependent decision rule that assigns an action $\mathbf{x}_t^\pi(\mathbf{s}_t) \in \mathcal{A}(\mathbf{s}_t)$ to every state $\mathbf{s}_t \in \mathcal{S}$ reached at the start of period $t$. Given an initial state $\mathbf{s}_1 \in \mathcal{S}$, the total expected reward of policy $\pi$ equals

$$\vartheta_1^\pi(\mathbf{s}_1) = \sum_{t \in \mathcal{T}} \mathbb{E}\left[r_t(\mathbf{s}_t^\pi, \mathbf{x}_t^\pi(\mathbf{s}_t^\pi))|\mathbf{s}_1\right],$$

where the expectation $\mathbb{E}$ is taken over the distribution of the random state $\mathbf{s}_t^\pi$ reached at the start of period $t$ when the planner selects actions according to policy $\pi$. Using $\Pi$ to represent the set of all feasible policies, our objective is to find an optimal policy $\pi^*$, that is,

$$\pi^*(\mathbf{s}_1) = \arg\max_{\pi \in \Pi} \vartheta_1^\pi(\mathbf{s}_1). \tag{2.1}$$

## 2.4. Approximate Linear Programming

In this section we describe our ALP approach to approximately solve the MDP (2.1). We first outline how MDP (2.1) can be formulated as a linear program in Section 2.4.1, and use this to construct an ALP formulation in Section 2.4.2 by imposing restrictions on the value function. In Section 2.4.3, we introduce reformulations of the ALP that can be solved efficiently and that attain an upper bound on the expected rewards. Finally, Section 2.4.4 describes how the solution to these ALP reformulations can be used to generate heuristic control policies that can provide a lower bound on the expected rewards; by comparing the performance of these control policies to the upper bounds, we can assess the quality of the resulting approximations.

### 2.4.1 Optimality Equations and Exact Linear Program

Given a feasible policy $\pi$, the value function $\vartheta_t^\pi(\mathbf{s}_t)$ can be defined recursively as

$$\vartheta_t^\pi(\mathbf{s}_t) = r_t(\mathbf{s}_t, \mathbf{x}_t) + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{x}_t) \vartheta_{t+1}^\pi(\mathbf{s}_{t+1}), \quad \forall t \in \mathcal{T}, \mathbf{s}_t \in \mathcal{S}.$$

Therefore, we can solve the MDP (2.1) and determine the optimal value function $\vartheta_t^*(\mathbf{s}_t)$ using the dynamic programming optimality equations

$$\vartheta_t^*(\mathbf{s}_t) = \max_{\mathbf{x}_t \in \mathcal{A}(\mathbf{s})} r_t(\mathbf{s}_t, \mathbf{x}_t) + \sum_{\mathbf{s}_{t+1} \in \mathcal{S}} P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{x}_t)\vartheta_{t+1}^*(\mathbf{s}_{t+1}), \quad \forall t \in \mathcal{T}, \mathbf{s}_t \in \mathcal{S}, \tag{2.2}$$

with boundary conditions $\vartheta_{T+1}^*(\mathbf{s}) = 0$ for all $\mathbf{s} \in \mathcal{S}$. Given the $|\mathcal{V}|$-dimensional state space and a $|\mathcal{C}|$-dimensional action space, however, solving (2.2) is computationally intractable due to the state-space and action-space explosion.

In this chapter, we therefore consider linear programming-based approximate dynamic programming approaches to solving this problem. As a first step, we use decision variables $\vartheta_t(\mathbf{s})$ for all $t \in \mathcal{T}$ and $\mathbf{s} \in \mathcal{S}$ to define an equivalent linear programming formulation for (2.2):

$$\begin{aligned} \mathbf{P}: \min_{\{\vartheta_t(\cdot)\}_{\forall t}} \quad & \vartheta_1(\mathbf{s}_1) \\ \text{s.t.} \quad \vartheta_t(\mathbf{s}) \geq \; & r_t(\mathbf{s}, \mathbf{x}) + \sum_{\mathbf{s}' \in \mathcal{S}} P(\mathbf{s}'|\mathbf{s}, \mathbf{x})\vartheta_{t+1}(\mathbf{s}'), \qquad \forall t \in \mathcal{T}, (\mathbf{s}, \mathbf{x}) \in \mathcal{X}. \end{aligned}$$

Solving ($\mathbf{P}$) is as difficult as solving (2.2), because both the number of columns and rows in ($\mathbf{P}$) are exponential in $|\mathcal{V}|$ and $|\mathcal{C}(L)|$. To address this challenge, a common strategy is to impose restrictions on the value function and *approximately* solve the LP. This is often achieved by expressing the value function $\vartheta_t(\mathbf{s})$ as the weighted sum of a set of basis functions; substituting the value function by this weighted sum in ($\mathbf{P}$) then yields an approximate linear program.

### 2.4.2 ALP Formulations

In this chatper, we focus on *affine* approximations of the value function. Affine approximations have been common in the LP-based ADP literature, and heuristic policies based on the affine approximation are known to give strong performance; see, for example, Adelman (2007), Vossen and Zhang (2015), and Meissner and Strauss (2012).

The affine value function approximation is given by

$$\vartheta_t(\mathbf{s}) \approx \theta_t + \sum_{i \in \mathcal{V}} V_{t,i} s_i, \tag{2.3}$$

where $V_{t,i}$ represents the marginal value of one type $i$ agent in period $t$ and $\theta_t$ is a constant offset. Substituting (2.3) into the LP formulation ($\mathbf{P}$) yields

$$\mathbf{P}^A : \min_{\theta,V} \; \theta_1 + \sum_{i\in\mathcal{V}} V_{1,i}s_{1,i}$$

$$\text{s.t.} \;\; \theta_t - \theta_{t+1} + \sum_{i\in\mathcal{V}} V_{t,i}s_i - \sum_{\mathbf{s}'\in\mathcal{S}} P(\mathbf{s}'|\mathbf{s},\mathbf{x}) \sum_{i\in\mathcal{V}} V_{t+1,i}s_i' \geq \sum_{c\in\mathcal{C}(L)} (1-\mu_c) r_{t,c}x_c - \sum_{i\in\mathcal{V}} w_{t,i}s_i,$$

$$\forall t\in\mathcal{T}, (\mathbf{s},\mathbf{x})\in\mathcal{X}.$$

For all $i\in\mathcal{V}$, we assume $\theta_{T+1}$ and $V_{T+1,i}$ equals 0 and observe that

$$\sum_{\mathbf{s}'\in\mathcal{S}} P(\mathbf{s}'|\mathbf{s},\mathbf{x})V_{t+1,i}s_i' = \mathbb{E}[s_i'|\mathbf{s},\mathbf{x}]V_{t+1,i} = \Big[(1-\nu_i)\big(s_i - \sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} (1-\mu_c)x_c\big) + \mathbb{E}[a_i]\Big]V_{t+1,i},$$

where $\mathbb{E}[a_i]$ denotes the expected number type $i$ arrivals in a period. This follows from the fact that both match failures and agent departures follow a binomial distribution.

For ease of exposition we define $\lambda_i = \mathbb{E}[a_i]$, $\overline{\nu}_i = 1-\nu_i$, and $\overline{\mu}_c = 1-\mu_c$ for all $i\in\mathcal{V}$ and $c\in\mathcal{C}(L)$. With these definitions, we can simplify ($\mathbf{P}^A$) to

$$\mathbf{P}^A : \min_{\theta,V} \; \theta_1 + \sum_{i\in\mathcal{V}} V_{1,i}s_{1,i}$$

$$\text{s.t.} \;\; \theta_t - \theta_{t+1} + \sum_{i\in\mathcal{V}} V_{t,i}s_i - \sum_{i\in\mathcal{V}} (\overline{\nu}_i s_i - \sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \overline{\nu}_i\overline{\mu}_c x_c + \lambda_i)V_{t+1,i} \geq \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c}x_c - \sum_{i\in\mathcal{V}} w_{t,i}s_i,$$

$$\forall t\in\mathcal{T}, (\mathbf{s},\mathbf{x})\in\mathcal{X}.$$

Because the affine approximation imposes a restriction on the value function, it immediately follows that the approximate linear program ($\mathbf{D}^A$) provides an upper bound on the total expected reward over the horizon. The dual of the affine ALP ($\mathbf{P}^A$) equals

$$\mathbf{D}^A: \max_{\{\pi_t(\cdot)\}_{\forall t\in\mathcal{T}}} \sum_{t\in\mathcal{T}} \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} \Big( \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c} x_c - \sum_{i\in\mathcal{V}} w_{t,i} s_i \Big) \pi_t(\mathbf{s},\mathbf{x})$$

$$\text{s.t.} \quad \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} \pi_t(\mathbf{s},\mathbf{x}) = 1, \qquad \forall t\in\mathcal{T}, \tag{2.4}$$

$$\sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} s_i \pi_t(\mathbf{s},\mathbf{x}) = \begin{cases} s_{1,i}, & \text{if } t=1, \\ \displaystyle\sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} (\overline{\nu}_i s_i - \sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \overline{\nu}_i \overline{\mu}_c x_c + \lambda_i) \pi_{t-1}(\mathbf{s},\mathbf{x}), & \text{if } t>1, \end{cases}$$

$$\forall i\in\mathcal{V}, t\in\mathcal{T}, \tag{2.5}$$

$$\boldsymbol{\pi}_t \geq \mathbf{0}, \qquad \forall t\in\mathcal{T}.$$

The decision variables $\pi_t(\mathbf{s},\mathbf{x})$ in ($\mathbf{D}^A$) can be viewed as *approximate* state-action probabilities. Specifically, $\pi_t(\mathbf{s},\mathbf{x})$ equals the fraction of time that the system reaches state $\mathbf{s}$ at the beginning of period $t$ and executes a matching $\mathbf{s}$ during the period under the affine approximation. These probabilities are approximate since they generally do not coincide with those following an optimal policy; see Adelman (2007) for further discussion. Since the term $\sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} s_i\pi_t(\mathbf{s},\mathbf{x})$ represents the expected number of type $i$ agents present at the start of period $t$, the flow balance constraints (2.5) maintain the state distribution in expectation over time.

Relative to ($\mathbf{P}$), the approximate linear program ($\mathbf{P}^A$) has a more manageable number of variables, although the number of constraints is still an exponential in both $|\mathcal{V}|$ and $|\mathcal{C}(L)|$. This suggests that the corresponding dual approximate linear program ($\mathbf{D}^A$) can be solved using column generation methods, which start with a subset of variables and iteratively add variables that can improve the solution. We refer to Trick and Zin (1997) for an early application of column generation methods for solving ALPs and to Desrosiers and Lübbecke (2005) for a detailed review of column generation methods in general.

### 2.4.3 ALP Reformulations

In this section, we introduce reformulations of the ALP dual ($\mathbf{D}^A$). The construction of such reformulations relies on the structure of the subproblems when using column generation to solve ($\mathbf{D}^A$). If we can formulate these subproblems as a linear program, the ALPs can be reformulated as well; see Vossen and Zhang (2015) for additional discussion. We investigate the conditions under which this leads to equivalent compact reformulations for ($\mathbf{D}^A$), and explore how this approach can be used to generate upper bounds by relaxing the ALP when the structure of the column generation subproblems does not admit an explicit characterization of the linear programming formulation.

As a first step, we consider the subproblems that arise when ($\mathbf{D}^A$) is solved with a column generation method. After solving a restricted master problem with a subset of the columns in ($\mathbf{D}^A$), suppose we obtain a dual solution $(\theta, V)$. Then, the column generation subproblem for period $t \in \mathcal{T}$ equals

$\mathbf{CG}_t$ :

$$\max_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c \left( r_{t,c} - \sum_{i\in\mathcal{V}(c)} \overline{\nu}_i V_{t+1,i} \right) x_c - \sum_{i\in\mathcal{V}} \left(V_{t,i} - \overline{\nu}_i V_{t+1,i} + w_{t,i}\right) s_i + \sum_{i\in\mathcal{V}} \lambda_i V_{t+1,i} - \theta_t + \theta_{t+1}.$$

Using the definition of the feasible state-action pairs, it follows that we can formulate the column generation subproblem as the **linear program**

$\mathbf{CG\text{-}LP}_t$ :

$$\max_{\hat{\mathbf{s}},\hat{\mathbf{x}}} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c \left( r_{t,c} - \sum_{i\in\mathcal{V}(c)} \overline{\nu}_i V_{t+1,i} \right) \hat{x}_c - \sum_{i\in\mathcal{V}} \left(V_{t,i} - \overline{\nu}_i V_{t+1,i} + w_{t,i}\right) \hat{s}_i + \sum_{i\in\mathcal{V}} \lambda_i V_{t+1,i} - \theta_t + \theta_{t+1}$$

$$\text{s.t.} \quad \sum_{c\in\mathcal{C}(L): i\in\mathcal{V}(c)} \hat{x}_c \le \hat{s}_i, \qquad \forall i \in \mathcal{V}, \tag{2.6}$$

$$\hat{s}_i \begin{cases} \le S, & \text{if } t > 1; \\ = s_{1,i}, & \text{if } t = 1. \end{cases} \qquad \forall i \in \mathcal{V}, \tag{2.7}$$

$$\mathbf{E}^c\hat{\mathbf{x}} + \mathbf{E}^s\hat{\mathbf{s}} \le \mathbf{e}, \tag{2.8}$$

$$\hat{\mathbf{x}}, \hat{\mathbf{s}} \ge \mathbf{0}.$$

Here, constraint (2.8) represents a collection of linear inequalities that, together with the other constraints in ($\mathbf{CG\text{-}LP}_t$), are necessary and sufficient to represent the convex hull of the feasible state-action pairs $(\hat{\mathbf{s}}, \hat{\mathbf{x}}) \in \mathcal{X}$.

Clearly, the nature of constraint (2.8) will be critical to our approach. First, however, we follow the approach taken in Vossen and Zhang (2015) to construct an equivalent reformulation for the approximate linear program ($\mathbf{D}^A$) based on the formulation ($\mathbf{CG\text{-}LP}_t$), as Proposition 2.1 states.

**Proposition 2.1.** ($\boldsymbol{D}^A$) *is equivalent to the linear program*

$$\boldsymbol{RD}^A : \max_{\hat{\mathbf{s}},\hat{\mathbf{x}}} \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c} \hat{x}_{t,c} - \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{V}} w_{t,i} \hat{s}_{t,i}$$

$$\text{s.t.} \qquad \hat{s}_{t,i} = \begin{cases} s_{1,i}, & \textit{if } t = 1; \\ \overline{\nu}_i \hat{s}_{t-1,i} - \sum\limits_{\substack{c\in\mathcal{C}(L): \\ i\in\mathcal{V}(c)}} \overline{\mu}_c \overline{\nu}_i \hat{x}_{t-1,c} + \lambda_i, & \textit{if } t > 1. \end{cases} \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}, \qquad (2.9)$$

$$\sum_{\substack{c\in\mathcal{C}(L): \\ i\in\mathcal{V}(c)}} \hat{x}_{t,c} \leq \hat{s}_{t,i}, \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}, \quad (2.10)$$

$$\mathbf{E}^c \hat{\mathbf{x}}_t + \mathbf{E}^s \hat{\mathbf{s}}_t \leq \mathbf{e}, \qquad \forall t \in \mathcal{T}, \quad (2.11)$$

$$\hat{\mathbf{x}}_t, \hat{\mathbf{s}}_t \geq 0, \qquad \forall t \in \mathcal{T}.$$

The ALP reformulation ($\mathbf{RD}^A$) can be viewed as a deterministic approximation of the original dynamic programming formulation. For each period $t \in \mathcal{T}$, constraints (2.9) track the system dynamics in expectation, while constraints (2.10) and (2.11) determine the matching in period $t$ based on the current state while maximizing the total expected reward.

Proposition 2.1 transforms an ALP formulation with an exponential number of variables ($\mathbf{D}^A$) into an ALP formulation with a potentially exponential number of constraints ($\mathbf{RD}^A$). This transformation relies on a complete linear programming description of the column generation subproblems ($\mathbf{CG}_t$) for all $t \in \mathcal{T}$, and extends the approach outlined in Vossen and Zhang (2015). While Vossen and Zhang (2015) derive compact equivalent reformulations for network revenue management problems, this application is substantially different from the dynamic stochastic matching problems we consider. Network revenue management problems assume at most one customer arrival in each period, while dynamic stochastic matching problems involve "block" demands for multiple agent types; as a result, the system dynamics and action space are significantly more complicated. In fact, the network revenue management problem can be interpreted as a special case of the dynamic stochastic matching problem, as we illustrate in Section A.2.

Of course, the tractability of the ALP reformulation ($\mathbf{RD}^A$) is highly dependent on the nature of the constraints in (2.11) that are needed to represent the convex hull of the feasible state-action pairs. While it may be hard to establish these constraints in general, there are important applications of dynamic matching that allow a full characterization of these constraints. Specifically, consider dynamic stochastic matching problems where the cycle length $L$ equals 2. As discussed before, this corresponds to settings where the action space can be represented as a general matching problem. For this case, the following proposition provides an explicit characterization of constraint (2.11) in ($\mathbf{RD}^A$).

**Proposition 2.2.** *For a dynamic matching instance defined on $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $L = 2$, ($\boldsymbol{RD}^A$) is equivalent to the linear program*

$$\max_{\hat{\mathbf{s}},\hat{\mathbf{x}}} \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c}\hat{x}_{t,c} - \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{V}} w_{t,i}\hat{s}_{t,i}$$

$$\text{s.t.} \qquad \hat{s}_{t,i} = \begin{cases} s_{1,i}, & \text{if } t = 1; \\ \overline{\nu}_i \hat{s}_{t-1,i} - \sum\limits_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(i)}} \overline{\mu}_c \overline{\nu}_i \hat{x}_{t-1,c} + \lambda_i, & \text{if } t > 1. \end{cases} \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}, \quad (2.12)$$

$$\sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \hat{x}_{t,c} \le \hat{s}_{t,i}, \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}, \quad (2.13)$$

$$\sum_{\substack{c\in\mathcal{C}(L):\\ \mathcal{V}(c)\in\mathcal{W}}} \hat{x}_{1,c} \le \frac{\sum_{i\in\mathcal{W}} s_{1,i} - 1}{2}, \qquad \forall \mathcal{W} \subseteq \mathcal{V} : \sum_{i\in\mathcal{W}} s_{1,i} \text{ odd}, \quad (2.14)$$

$$\hat{\mathbf{x}}_t, \hat{\mathbf{s}}_t \ge 0, \qquad \forall t \in \mathcal{T}.$$

The resulting ALP formulation when $L = 2$ has an exponential number of constraints (2.14).

We observe that a **compact** equivalent formulation is in fact unattainable in this case: when $T = 1$ the formulation reduces to a static $b$-matching problem, and Rothvoß (2017) shows that no polynomial-sized formulation can represent the convex hull of feasible matchings. However, it is known that the blossom inequalities (2.14) can be separated efficiently (Padberg and Rao 1982). Moreover, the number of such constraints that need to be added is small in our applications; as a result, our numerical study indicates that solving ($\mathbf{RD}^A$) is orders of magnitude faster than solving the approximate linear program ($\mathbf{D}^A$) using column generation methods.

In addition, we observe that constraint (2.14) is redundant when the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is bipartite. Dynamic stochastic matching problems with a bipartite graph structure arise in variety of applications, such as ridesharing and online advertising. Solving the approximate linear programs is therefore highly efficient in these applications, as we show for the dynamic ridesharing instances we consider in our numerical study.

Proposition 2.2 no longer applies when the cycle lengths $L$ are restricted to some value other than 2. In fact, when $L \geq 3$ it is generally impossible to fully characterize the convex hull of the feasible state-action pairs expressed in constraints (2.11). This follows from Theorem 1 in Abraham et al. (2007), which shows that the barter exchange clearing problem is NP-Complete given a graph $G = (\mathcal{V}, \mathcal{E})$ and an integer $L \geq 3$. As a result, the column generation subproblems ($\mathbf{CG}_t$) for ($\mathbf{D}^A$) will also be NP-Complete, and therefore the constraints that specify the convex hull of the feasible state-action pairs cannot easily be characterized or separated. In such cases, however, we can still consider a **relaxation** of the approximate linear program ($\mathbf{D}^A$) by removing the constraints (2.11). Because this relaxed ALP will yield an upper bound on the affine approximation, it also provides a bound on the total expected reward. Such relaxations can still provide strong performance if the column generation subproblem relaxations ($\mathbf{CG\text{-}LP}_t$) without constraint (2.11) have tight bounds. In our numerical study, we pursue this approach for a class of dynamic stochastic matching problems that arises in kidney exchange. It might further be possible to strengthen this relaxed ALP by adding valid inequalities for the barter exchange clearing problem. However, our numerical experiments indicate that the benefits of such valid inequalities would be marginal, and we do not consider this potential bound strengthening in this chapter.

In sum, the ALP reformulation ($\mathbf{RD}^A$) provides an efficient approach to generate upper bounds on the total expected reward for dynamic stochastic matching problems. For bipartite matching instances, ($\mathbf{RD}^A$) has $\mathcal{O}(T \cdot (|\mathcal{V}| + |\mathcal{C}|))$ variables and $\mathcal{O}(T \cdot |\mathcal{V}|)$ constraints, hence both the space and time complexity of ($\mathbf{RD}^A$) are polynomial. For general non-bipartite instances with $L = 2$, ($\mathbf{RD}^A$) has $\mathcal{O}(T \cdot (|\mathcal{V}| + |\mathcal{C}|))$ variables, $\mathcal{O}(T \cdot |\mathcal{V}|)$ *regular* constraints and exponentially many *blossom* constraints. However, because the blossom constraints are added to the formulation in a dynamic fashion with efficient separation routines, the space and time complexity of ($\mathbf{RD}^A$) remain polynomial. For general instances with $L \geq 3$, we remove constraints (2.11) from the formulation and solve a *relaxed* version of ($\mathbf{RD}^A$), which also has $\mathcal{O}(T \cdot (|\mathcal{V}| + |\mathcal{C}|))$ variables and $\mathcal{O}(T \cdot |\mathcal{V}|)$ constraints.

### 2.4.4 Heuristic Control Policies

In addition to providing an upper bound, optimal solutions for the ALP reformulations can also be used to derive heuristic control policies. We distinguish between **primal** and **dual** control policies.

Primal policies rely on an an optimal solution $(\boldsymbol{V}^*, \boldsymbol{W}^*, \boldsymbol{U}^*)$ to the primal formulation ($\mathbf{RP}^A$) that corresponds to ($\mathbf{RD}^A$), which we do not state explicitly. Observe that the values of the decision variables $V^*_{t,i}$ reflect the marginal value of a type $i$ agent in period $t$. Based on these values, we can derive a one-step greedy policy by substituting the approximation (2.3) that results with these values into the right hand side of (2.2) for each initial state $\mathbf{s}_1 \in \mathcal{S}$ (Powell 2011, Chapter 6). Disregarding constant terms in the objective function, this results in the optimization problem

$$\max_{\mathbf{x} \in \mathcal{A}(\mathbf{s}_1)} \sum_{c \in \mathcal{C}(L)} \bar{\mu}_c \left( r_{1,c} - \sum_{i \in \mathcal{V}(c)} V^*_{2,i} \bar{\nu}_i \right) x_c.$$

We can choose to implement this policy without or with resolving. Without resolving we solve the ALP ($\mathbf{RD}^A$) once, and the value function estimates used in the policy do not change. With resolving, we solve ($\mathbf{RD}^A$) after every state update and update the value function estimates in the objective function.

In addition, we also consider control policies that are based on the optimal solution to the dual program ($\mathbf{RD}^A$). This naturally leads to probabilistic allocation policies, which have shown promise in several applications (Lei et al. 2021, Vossen et al. 2022). To illustrate this approach, suppose $(\mathbf{s}^*, \mathbf{x}^*)$ is an optimal solution to ($\mathbf{RD}^A$) when $L = 2$. We focus on the first period variables $\mathbf{x}_1^*$, and observe that $\mathbf{x}_1^*$ might be fractional due to linkages present in (2.9) (even though the column generation subproblems ($\mathbf{CG\text{-}LP}_t$) for a single period would have integer solutions). The following proposition states that $\mathbf{x}_1^*$ can be expressed as a convex combination of feasible integer matching decisions.

**Proposition 2.3.** *Let $\mathbf{x}_1^*$ be the first period solution of an optimal solution to ($\boldsymbol{RD}^A$) when $L = 2$. Then,*

$$\mathbf{x}_1^* = \sum_{n=1}^{N} \rho_n \mathbf{x}_1^n, \tag{2.15}$$

*where $\rho_n \geq 0$ for $n = 1, ..., N$ and $\sum_{n=1}^{N} \rho_n = 1$, and $\mathbf{x}_1^n$ is a feasible integer solution of the first period for $n = 1, ..., N$.*

The proof of Proposition 2.3 relies on an extension of the Birkhoff-von Neumann Theorem proposed recently in Vazirani (2020). Given this decomposition, we obtain a probabilistic allocation policy by selecting the matching $\mathbf{x}_1^n$ with probability $\rho_n$. To use this approach, we have to update and solve ($\mathbf{RD}^A$) at the start of every period. When $L \geq 3$, Proposition 2.3 no longer applies. To construct an allocation policy in this setting, we solve a relaxation of ($\mathbf{RD}^A$) by removing constraints (2.11) and round down any fractional solution; clearly, this yields a feasible integer allocation.

We use Monte Carlo simulation to evaluate these policies, and generate sample paths that advance the state by sampling from the stochastic process that describes the uncertain state transitions. Starting from an initial state, we calculate the sample average reward (and standard errors) of these policies over the horizon to determine a lower bound on the policies' performance. This might require resolving the ALP after each state update. The resulting lower bound can be compared with the upper bound obtained by solving the ALP reformulation to assess overall quality. Section A.3 provides additional implementation details for the policies outlined above.

## 2.5. Numerical Study

In this section, we evaluate the performance of our approach with a numerical study that considers three different applications. We compare the performance of our ALP approach to that of several benchmark policies, which we outline in Section 2.5.1. Section 2.5.2 considers real-time ridesharing problems, which involve determining a bipartite matching in each period. Section 2.5.3 considers matchmaking problems in on-line video games, which involve matchings in general graphs. Section 2.5.4 evaluates the performance of our approach for dynamic kidney exchange problems, which involve matching patient-donor pairs for kidney transplants and allows for longer exchange cycles.

The ALP and all benchmark approaches are implemented using `Python 3.8.5`, and we use the `Gurobi` solver whenever a linear or integer program is solved. All code and data sets are available at `https://github.com/FanYouCN/dynamic_stochastic_matching`. We used a PC with AMD Ryzen 3800X and 16GB of RAM running Windows 10 to conduct our experiments.

### 2.5.1 Benchmark Approaches

We use three benchmarks to assess the performance of our ALP reformulations: column generation, a deterministic linear program, and a limited lookahead policy. We outline these benchmarks below, while Section A.3 provides additional details on their implementations.

***Column Generation.*** We first compare our ALP reformulation with solving ($\mathbf{D}^A$) using column generation. In particular, we iteratively add columns for each period $t \in \mathcal{T}$ by solving the subproblems ($\mathbf{CG}_t$) as an IP. The purpose of this comparison is twofold. First, we compare computation times to demonstrate the efficiency of the ALP reformulation. In addition, we compare upper bounds obtained by both approaches to understand the impact of relaxing the ALP when a complete description of constraint (2.11) is unattainable.

***Deterministic Linear Program.*** We also compare the performance of our ALP with a deterministic linear program (DLP) that assumes away all uncertainty and replaces relevant terms by their respective expected values. DLPs have received considerable attention, see for example Chapter 2 of Gallego et al. (2019). While DLPs typically aggregate time periods, our setting (which includes potential failures and departures) requires a multi-stage LP formulation. The DLP not only provides an upper bound, but also admits the derivation of heuristic control policies akin to the primal policies outlined in Section 2.4.4.

***Limited Lookahead Policy.*** Finally, we also contrast the performance of our ALP approach with limited lookahead policies (LLA). Limited lookahead policies optimize matching decisions over some pre-determined horizon, which is usually significantly shorter than $\mathcal{T}$. A horizon length of zero periods yields a myopic policy that only maximizes immediate rewards without accounting for future dynamics. In our numerical experiments, we set the horizon length to 5 periods. For a more detailed discussion of myopic and limited lookahead policies, we refer to Chapter 6 of Powell (2011).

### 2.5.2 Dynamic Matching for Ridesharing

We first assess the performance of our approach for dynamic stochastic matching problems that arise in ridesharing platforms, which involve matchings over a bipartite compatibility graph. We perform an initial set of experiments based on an example in Özkan and Ward (2020, p. 25); subsequently, we also evaluate our approach on randomly generated larger instances to evaluate the scalability of our approach.

#### 2.5.2.1 Initial Experiments

For our initial set of experiments, we adapt an example from Özkan and Ward (2020), where the ridesharing region is partitioned into 3 disjoint areas. Area 1 only has customer arrivals while area 3 only has driver arrivals, causing an imbalance between supply and demand in these regions. Different instances are generated by varying a parameter $n$ that captures the *thickness* of the

Table 2.1: Computation times (seconds) of instances in Ozkan and Ward (2020)

| | n=1 | n=10 | n=100 |
|---|---|---|---|
| ALP | $1.6 \times 10^{-2}$ | $5.9 \times 10^{-2}$ | $3.5 \times 10^{-1}$ |
| ALP_CG | $3.6 \times 10^{-1}$ | $1.5 \times 10^{1}$ | $9.8 \times 10^{2}$ |
| DLP | $7.7 \times 10^{-3}$ | $4.7 \times 10^{-2}$ | $2.2 \times 10^{-1}$ |
| OLP | | $4.4 \times 10^{-3}$ | |

matching market. Specifically, $n$ represents the arrival rate of customers and drivers at each of the locations. For a detailed description of the instances derived from this example, we refer to Section A.4.

In addition to the ALP-based and benchmark policies we also assess the performance of the myopic LP in Özkan and Ward (2020, Section 4.1), which we refer to as OLP. The OLP ignores any state information and market thickness while matching arriving customers and drivers in expectation. The OLP provides an upper bound, while its optimal solution can be used to construct a randomized policy.

Table 2.1 summarizes the computation times when solving the ALP reformulation, the ALP using column generation, the DLP, and the OLP. These results highlight the computational efficiency of our ALP reformulation, which solves orders of magnitude faster than the column generation methods. Except for the OLP, we also observe that computation times increase slightly as the market thickness $n$ increases. This is explained by the fact that discretizing the time horizon requires a larger number of periods when the market thickness increases (see Section A.4 for details on this discretization).

Table 2.2 summarizes the performance of the ALP-based and benchmark control policies. For each thickness level, we first determine upper bounds. Next, we generate 100 sample paths of arrivals and simulate different policies. Table 2.2 reports the sample average number of matchings (i.e., the reward), the standard errors of the sample averages, and the average optimality gaps. When determining optimality gaps, we compare the ALP and LLA policies against the ALP bound, the DLP policy against the DLP bound, and the OLP policy against the OLP bound. For these initial instances, we observe that the ALP policies, the DLP policy, and the OLP policy all

Table 2.2: Upper and lower bounds of instances in Ozkan and Ward (2020)

| | | n=1 | n=10 | n=100 |
|---|---|---|---|---|
| ALP UB | | 8.86 | 97.5 | 984.1 |
| DLP UB | | 10.0 | 100.0 | 1000.0 |
| OLP UB | | 9.85 | 98.5 | 985.1 |
| ALP Dual Policy | lb(se) | 5.53(0.2) | 81.8(0.81) | 936.6(2.5) |
| | gap | 37.6 | 16.1 | 4.8 |
| ALP Primal Policy with resolve | lb(se) | 5.59(0.2) | 83.65(0.8) | 945.0(2.6) |
| | gap | 36.9 | 14.2 | 3.9 |
| ALP Primal Policy no resolve | lb(se) | 5.53(0.2) | 83.64(0.8) | 944.8(2.6) |
| | gap | 37.6 | 14.2 | 4.0 |
| DLP Policy | lb(se) | 5.60(0.2) | 82.93(0.8) | 943.0(2.8) |
| | gap | 44.0 | 17.0 | 5.7 |
| OLP Policy | lb(se) | 5.58(0.2) | 82.79(0.8) | 941.1(2.7) |
| | gap | 43.3 | 15.9 | 4.5 |
| LLA Policy | lb(se) | 5.52(0.2) | 72.9(0.8) | 771.3(2.6) |
| | gap | 37.3 | 25.2 | 21.6 |

have similar performance, and significantly outperform the LLA policy. However, the ALP upper bound is stronger across all thickness levels and allows us to establish the tightest gaps. The gap improvement relative to the OLP decreases as thickness levels increase, which is consistent with the asymptotic optimality provided by the OLP upper bound in Özkan and Ward (2020). Overall, these results highlight both the efficiency and quality of our ALP-based approach relative to the benchmark policies. While the OLP policy is slightly more efficient, we emphasize that this critically depends on the fact that all parameters in these initial instances are time-homogeneous: with time-varying arrival rates, the continuous LP becomes significantly more challenging; see the discussion at the end of Section 3 in Özkan and Ward (2020). In contrast, this will have little impact on solving the ALP reformulation.

#### 2.5.2.2 Experiments with Larger Ridesharing Instances

To further assess the performance of our ALP-based policies, we also generate larger random instances that incorporate a richer reward structure and allow for latency in the matching decisions. By slightly increasing potential latency of matching decisions, the platform accumulates a *batch* of customer requests, resulting in a more efficient assignment of the drivers. Due to their substantial benefits, such batching algorithms have seen widespread adoption in major ridesharing platforms. For example, Zhang et al. (2017) describe the batching algorithm used by DiDi, and an illustration of the batching used by Uber can be found in Uber (2019). We provide a detailed description of these instances in Section A.4.

As before, we evaluate the performance of the ALP-based and benchmark policies on these instances. However, we do not include the OLP policy because it cannot accommodate batch requests. A.4 lists average computation times over a broad set of larger instances. For these larger instances, we first observe that solving the ALP reformulation remains efficient while solving the ALP using column generation quickly becomes intractable; this further demonstrates the value of the reformulation.

Table 2.3: Upper and lower bounds for ridesharing instances with 20 areas

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 1254.9 | 1202.1 | 1199.6 | 1204.6 | 1108.0 |
| DLP UB | | 1348.7 | 1304.6 | 1293.8 | 1305.0 | 1227.7 |
| ALP Dual Policy | lb(se) | 1241.7(2.2) | 1192.2(2.6) | 1184.6(1.6) | 1192.6(2.0) | 1102.4(2.5) |
| | gap | 1.05 | 0.82 | 1.25 | 0.82 | 0.50 |
| ALP Primal Policy with resolve | lb(se) | 1241.6(2.2) | 1191.8(2.6) | 1184.5(1.7) | 1192.3(2.0) | 1102.1(2.5 |
| | gap | 1.06 | 0.85 | 1.26 | 1.01 | 0.53 |
| ALP Primal Policy no resolve | lb(se) | 1233.5(2.3) | 1184.3(2.6) | 1177.5(1.7) | 1183.9(2.0) | 1096.0(2.4) |
| | gap | 1.70 | 1.47 | 1.84 | 1.72 | 1.08 |
| DLP Policy | lb(se) | 1104.0(2.2) | 1091.4(2.4) | 1101.1(1.7) | 1079.4(1.8) | 1018.0(2.2) |
| | gap | 18.1 | 16.3 | 14.9 | 17.3 | 17.1 |
| LLA Policy | lb(se) | 1193.2(2.1) | 1169.2(2.4) | 1149.6(1.7) | 1161.0(1.9) | 1083.2(2.4) |
| | gap | 4.91 | 2.73 | 4.16 | 3.61 | 2.24 |

Table 2.3 shows the policy performance for five randomly generated ridesharing instances with 20 areas; again, we show upper bounds, sample average rewards and standard errors as well as optimality gaps. We observe that the ALP bound is significantly tighter than the DLP bound across all instances. Both the ALP primal policy and the ALP dual policy with resolve show near optimal performance, and consistently outperform the DLP and LLA policies; The ALP primal policy without resolve also shows strong performance, and might be an attractive alternative for large-scale problems where policies that require resolving become too time-consuming for real-time

decision-making. In A.4 we show results for an additional collection of instances, which exhibit similar performance characteristics. Overall, these experiments show that our approach provides high quality bounds and control policies for dynamic stochastic matching problems over bipartite graphs.

### 2.5.3 Matchmaking in Online Video Games

In this section, we evaluate the performance of our ALP approach for matchmaking problems that arise in online video games. In contrast to ridesharing problems, matchmaking involves determining matchings over a general non-bipartite graph. In addition, this setting also includes *self loops* in the compatibility graphs, in that players with an identical type can potentially be matched. The key difficulty in the 1-versus-1 (1v1) games we consider is balancing the quality of the match that occurs when two players are paired with the time players spend waiting for a match to begin. While an important area of research in and of itself, we assume the quality of a pairing can be represented using a “score” function. We provide further background in A.4, and refer to van Dongen (2018) for an illustrative example .

#### 2.5.3.1 Initial Experiments

We randomly generate 5 baseline instances by considering the matchmaking system for a 1v1 online video game in the US. Players are divided into four geographical regions, *west*, *midwest*, *south*, and *northeast*. We assume that each region has dedicated servers to host the online game sessions. Matching players from the same region will result in a lower latency, matching players from adjacent regions leads to higher latency, while players from the west and northeast regions are incompatible and cannot be matched. Players are divided into three skill level brackets, *low*, *mid*, and *high*, and generally prefer to be matched with other players of the same skill level. Thus, matching players from the same bracket results in a higher reward, while low skilled players are incompatible with high skilled players. Each of the 5 baseline instances has a unique set of parameters that capture the reward, waiting cost, failure rate, and arrival rate; we refer to A.4 for a detailed description of

Table 2.4: Upper and lower bounds for baseline matchmaking instances

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 3425.1 | 3642.5 | 3385.1 | 2952.4 | 3736.2 |
| DLP UB | | 4087.8 | 4356.5 | 4020.9 | 3505.3 | 4441.2 |
| | lb(se) | 3203.7(17.3) | 3419.2(17.0) | 3139.5(18.1) | 2700.5(16.8) | 3533.8(16.5) |
| ALP Dual Policy | gap | 6.46 | 6.13 | 7.25 | 8.53 | 5.41 |
| ALP Primal Policy with resolve | lb(se) | 3225.7(16.4) | 3431.8(18.3) | 3140.5(17.3) | 2704.0(16.9) | 3523(17.8) |
| | gap | 5.82 | 5.78 | 7.22 | 8.41 | 5.68 |
| ALP Primal Policy no resolve | lb(se) | 3101.2(23.1) | 3407.9(18.9) | 2850.1(34.5) | 2638.5(16.9) | 3477.0(19.0) |
| | gap | 9.45 | 6.43 | 15.8 | 10.6 | 6.93 |
| | lb(se) | 3199.1(17.6) | 3402.7(17.4) | 3141.7(18.2) | 2689.7(15.5) | 3508.8(17.9) |
| DLP Policy | gap | 21.7 | 21.9 | 21.8 | 23.2 | 21.0 |
| | lb(se) | 3057.3(15.2) | 3304.5(16.7) | 3015.3(17.1) | 2500.6(15.3) | 3466.2(18.3) |
| LLA Policy | gap | 10.7 | 9.28 | 10.9 | 15.3 | 7.22 |

these parameters.

As a first step, we evaluate the computation times when solving the ALP reformulation, the ALP using column generation, and the DLP. On average, solving the DLP requires 0.09 seconds, while solving the ALP reformulation requires 0.19 seconds, and solving the ALP using column generation requires 116 seconds. In this setting, solving the ALP reformulation uses a separation procedure to dynamically add the blossom inequalities (2.14). Nevertheless, the ALP reformulation is again orders of magnitude faster than the column generation, further highlighting the computational efficiency of our framework.

Next, we assess the performance of the ALP-based and benchmark control policies. As before, we generate 100 sample paths and simulate the different control policies. We again compare the ALP and LLA policies against the ALP bound, and the DLP policy against the DLP bound. Table 2.4 summarizes the results. We observe that the ALP upper bound is tighter than the DLP upper bound, which allows us to establish significantly stronger optimality gaps. The ALP dual policy and primal policy with resolving show similar performance, and slightly outperform the DLP policy and the ALP primal policy without resolve. The LLA policy performs markedly worse than all other policies. Generally speaking, these results are in line with our observations for the ridesharing instances, and illustrate the broad applicability of our framework.

#### 2.5.3.2 Further Experiments

To understand how our ALP approach performs in different environments, we randomly generate a number of additional instances by systematically varying the waiting costs, arrival rates, and sizes of the underlying compatibility graphs. A description and the results of these additional experiments can be found in A.4 Different environments lead to certain broad qualitative changes to the results in these additional experiments. For example, the relative performance of the LLA policy does improve when waiting costs are low or when arrival rates increase, as it is easier to pair players with a perfect opponent in such cases. Nevertheless, the key observations from our initial experiments remain valid. The ALP bound is still uniformly stronger in all experiments, and the ALP dual and primal policy with resolve predominantly yield the highest rewards. Overall therefore, our model performs well across different settings, and achieves strong performance on general non-bipartite compatibility graphs. To the best of our knowledge, our model is the first to consider dynamic matchmaking decisions with stochastic player arrivals and departures.

### 2.5.4 Kidney Exchange

In our final set of experiments, we consider the performance of our ALP framework for dynamic kidney exchange problems. Dynamic kidney exchange problems incorporate a broad set of distinct considerations: matchings can be defined over general non-bipartite graph (with self-loops), or even incorporate longer exchange cycles. Furthermore, these problems have stochastic arrivals and departures, and include the possibility of match failures.

We generate instances for this problem using the well-known *Saidman Generator* (Saidman et al. 2006), where patient-donor pairs are characterized by various attributes. We follow the approach outlined in Dickerson et al. (2018) and Li et al. (2019), and a detailed description can be found in A.4.

We first evaluate computation times when solving the ALP reformulation and the DLP, using cycle length restrictions $L = 2$ and $L = 3$. When $L = 2$, the ALP reformulation again uses a

Table 2.5: Upper and lower bounds for kidney exchange instances with two-way exchanges

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 112860.3 | 122342.2 | 117266.3 | 113375.9 | 120089.9 |
| DLP UB | | 127403.2 | 136072.6 | 129362.7 | 129131.4 | 134752.7 |
| | lb(se) | 110123.2(325.3) | 119673.4(340.6) | 113901.4(345.8) | 110922.9(306.3) | 116890.0(343.2) |
| ALP Dual Policy | gap | 2.42 | 2.18 | 2.86 | 2.16 | 2.66 |
| | lb(se) | 110168.2(321.9) | 119735.5(340.8) | 113952.1(345.9) | 110985.9(304.5) | 116990.3(342.0) |
| ALP Primal Policy with resolve | gap | 2.38 | 2.13 | 2.82 | 2.10 | 2.58 |
| | lb(se) | 110073.6(323.2) | 119686.7(344.2) | 113821.9(345.1) | 110863.2(307.7) | 116741.3(340.6) |
| ALP Primal Policy no resolve | gap | 2.46 | 2.17 | 2.93 | 2.21 | 2.78 |
| | lb(se) | 98778.9(324.6) | 108406.5(325.9) | 103959.1(348.3) | 101134.3(309.8) | 107368.0(339.7) |
| DLP Policy | gap | 22.4 | 20.3 | 19.6 | 21.6 | 20.3 |
| | lb(se) | 109386.2(326.6) | 118980.6(343.4) | 113234.7(348.4) | 110066.7(309.8) | 116158.0(344.8) |
| LLA Policy | gap | 3.07 | 2.74 | 3.43 | 2.91 | 3.27 |

separation procedure to dynamically add the blossom inequalities (2.14). When $L = 3$, we solve a relaxed ALP as described in Section 2.4.3. On average, solving the ALP requires 1.6 seconds when $L = 2$ and 51.1 seconds when $L = 3$. Solving the DLP requires 1.1 seconds when $L = 2$ and 44.5 seconds when $L = 3$. Solving the ALP using column generation is intractable for these instances. To assess the loss of quality in the bounds obtained when solving the relaxed ALP for $L = 3$, we therefore also generated smaller instances. We refer to A.4 for a description of these instances, and a summary of the results when solving these smaller instances using both the relaxed ALP and the ALP with column generation. Overall, these results indicate that removing constraints (2.11) in the relaxed ALP has a negligible impact on the quality of the upper bounds. Moreover, we again observe that the ALP reformulation solves orders of magnitude faster than the ALP using column generation.

We generate 5 kidney exchange instances, where edge rewards are randomly drawn. For each instance, after determining the ALP and DLP upper bounds, we generate 100 sample paths and simulate the various policies. Tables 2.5 and 2.6 summarize the performance of the ALP-based and benchmark control policies. As with the previous applications, the ALP upper bound is considerably stronger than the DLP upper bound and the ALP-based policies provide the highest rewards. Overall, our ALP approach obtains near-optimal performance, even when using the relaxed ALP for $L = 3$. Moreover, a comparison of the upper and lower bounds for $L = 2$ and $L = 3$ shows that the improvement by allowing three-way exchanges is substantial ($\sim 5\%$), which is consistent

Table 2.6: Upper and lower bounds for kidney exchange instances with three-way exchanges

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 116894.2 | 127697.0 | 123204.3 | 114750.9 | 124055.0 |
| DLP UB | | 135670.1 | 146427.1 | 139618.3 | 133642.1 | 141178.8 |
| ALP Dual Policy | lb(se) | 114514.8(339.6) | 124794.3(373.7) | 120153.7(310.3) | 112218.7(293.8) | 121795.3(336.0) |
| | gap | 2.03 | 2.27 | 2.47 | 2.20 | 1.82 |
| ALP Primal Policy with resolve | lb(se) | 114784.2(340.2) | 125067.1(370.8) | 120362.5(311.8) | 112475.1(299.3) | 122032.7(333.9) |
| | gap | 1.80 | 2.05 | 2.30 | 1.98 | 1.63 |
| ALP Primal Policy no resolve | lb(se) | 114500.2(348.1) | 124758.3(379.5) | 120066.1(305.2) | 112208.3(301.5) | 121804.0(331.3) |
| | gap | 2.04 | 2.30 | 2.54 | 2.21 | 1.81 |
| DLP Policy | lb(se) | 101322.2(334.3) | 111720.6(354.8) | 106728.6(309.7) | 99654.3(279.2) | 110584.8(335.0) |
| | gap | 25.3 | 21.6 | 23.7 | 25.4 | 21.6 |
| LLA Policy | lb(se) | 113540.8(349.9) | 123817.0(362.1) | 119285.4(311.3) | 111079.6(299.8) | 120888.5(340.1) |
| | gap | 2.86 | 3.03 | 3.18 | 3.19 | 2.55 |

with other experiments in the literature. These results demonstrate the effectiveness of our ALP approach for dynamic kidney exchange.

## 2.6. Concluding Remarks

We introduce a general framework for dynamic stochastic matching problems that provides a unifying alternative to different approaches that have been proposed for various applications that involve matching over time. Our framework incorporates common features such as failure-awareness and arbitrary length cycles, which arise in a variety of applications. Using the linear programming-based approximate dynamic programming approach, we introduce novel reformulations of the resulting ALPs that can be solved efficiently even if they are not necessarily compact. Even when our ALP reformulations are not equivalent and relax the original ALPs, numerical experiments illustrate that our approach achieves strong performance and provides an attractive alternative for applications that involve dynamic stochastic matching.

In practice, different applications naturally admit additional unique considerations. For example, agents' departure rates might differ based on whether they encountered a match failure. In dynamic ridesharing, joint pricing and matching could lead to better overall efficiency. We could also consider carpooling, where the system controller matches multiple customers to one driver. For kidney exchange, fairness consideration could be an important extension. It would be interesting to see how our model can be adjusted to account for such scenarios.

Even though the affine approximation already yields high quality performance, it could also be of interest to consider alternative approximation schemes. For example, a separable piecewise linear approximation might potentially lead to stronger bounds and better heuristic policies. A key challenge, however, is to retain tractability when solving the approximate linear programs that would arise from such approximation schemes.

# Chapter 3

# Inventory Aware Customer Choice Modeling

**Abstract.** We study the customer choice modeling problem faced by online centralized platforms that sell *virtual goods*. Examples of such platforms include item stores of online multiplayer video games and certain marketplaces in the *metaverse*. One key distinctive feature of these platforms is the access to complete purchase history and *inventory*[1] information of their customers, in contrast to online marketplaces that sell physical goods. We aim to understand customers' choice behaviours in the presence of perfect inventory information, and propose *inventory aware* discrete choice models that incorporate this information. We conduct extensive numerical experiments by first setting up so-called ground truths, which are used to simulate inventory aware choice models and other benchmark approaches. Our results show that inventory aware models can substantially improve choice model performance both in terms of predictive power and revenue.

## 3.1. Introduction and Related Work

Discrete choice models are used to model choices made by customers among a finite set of alternatives. Accounting for discrete choice behavior has received increased attention in recent years, as key operational decisions such as assortment planning and pricing require high-quality inputs.

Traditionally, operational decisions have been based on demand models that are centered

[1]Following video game terminology, inventory refers to the customers' ownership information of virtual items throughout this work.

around products, resulting in the so-called *independent demand paradigm*. Under this paradigm, each product has its own stream of demand, independent of other products offered. The independent assumption is clearly somewhat unrealistic. For example, in revenue management problems the probability of selling a full-fare ticket may likely depend on whether a discounted ticket is also available at the same time.

The importance of accounting for such customer choice behaviours has been recognized by both researchers and practitioners. The seminal work of Talluri and Van Ryzin (2004) demonstrated the importance of incorporating customer choice behavior into revenue management models. Since their formulation of the single-leg airline seat allocation problem under a general discrete choice model, there has been a large amount of research on choice based revenue management models. Several studies have empirically reported the success of choice based revenue management models. For example, Yunes et al. (2007) describe how John Deere & Co. identified a potential increase of profit from 8% to 18% by investigating consumer choice preferences for two of its product lines. In Lee et al. (2016), the authors conduct a choice based demand estimation at a university bookstore and evaluate the impact of using these demand estimates to determine the inventory poldicy. They implemented the proposed model, which increased profits by more than 10% in a controlled field experiment. The article by Vulcano et al. (2010) reports the results of a study conducted with a commercial airline. The authors estimate a multinomial logit choice model and access the revenue performance of a choice model based policy against an independent demand model based policy in a simulation study, resulting in revenue improvements between 1.4% an 5.3%.

Given the documented success of accounting for consumer choice behaviors, a stream of work has emerged that focuses on better understanding the operational impact of customer choices. However, the study of customer choice behaviours itself already dates back to the 1950s in the field of psychology and economics, see Luce (1959); since then, various discrete choice models have been proposed. To date, the most popular discrete choice models are the multinomial logit (MNL, see McFadden et al. 1973) model and its extensions, including mixed multinomial logit (MMNL, see McFadden and Train 2000), latent-class MNL (e.g., Greene and Hensher 2003) and nested logit (NL, e.g., McFadden 1981). The MNL model and its extensions belong to a class of *utility-based models*, where customers obtain a utility when choosing an alternative and their choices are governed by a certain decision criterion, e.g., utility maximization. The specific structures of choice models in this class depend on assumptions relative to the random utility distributions that are used. Another example of random utility based choice model is the exponomial model (see, e.g., Alptekinoğlu and Semple 2016). Other classes of discrete choice models include *temporal models*, e.g., the Markov chain choice model proposed in Blanchet et al. (2016), and *rank-based models* such as the ranked-list choice models in Van Ryzin and Vulcano (2015), van Ryzin and Vulcano (2017). For a comprehensive review of discrete choice models, we refer to Feng et al. (2022), where the authors also discuss relationships among different classes of structural choice models. For a empirical evaluation of discrete choice models, we refer to Berbeglia et al. (2021).

As mentioned earlier, discrete choice models have been successfully applied in the field of product line optimization (Yunes et al. 2007), traditional retail (Kök and Fisher 2007, Kök et al. 2008), airline revenue management (Zhang and Cooper 2005, Vulcano et al. 2010) and E-commerce retailers (Rusmevichientong et al. 2010, Bernstein et al. 2019). However, decision makers in different application domains have different levels of information that can be used to understand and estimate customers' choices. Here, we focus our attention on a centralized online marketplace that sells virtual goods and where the seller has access to **perfect** information for each customer; specifically, the seller knows each customer's complete purchase history and inventory. One example of such marketplaces is the virtual item store in online multiplayer games, which is a large and growing industry with revenues in excess of a billion dollars (Newzoo 2020). Another potential example can be found in the marketplaces within Meta's (formally Facebook) vision of the metaverse; although many believe that the metaverse should be built upon decentralized block-chain technology, a Meta-controlled metaverse would likely result in centralized storage of user data, providing the opportunity to more accurately estimate customers' choice preferences.

The unique characteristics of these online marketplaces, which involve virtual goods and access to purchase history and inventory information, present novel challenges for customer choice modeling. To illustrate this, it is instructive to compare the item store of Call of Duty Warzone (an online video game with over 120 million players) with an e-retailer such as Amazon.com. In Warzone, the seller has every customer's ownership information of every possible in-game virtual item. Amazon, on the other hand, may never realistically know whether its customers already own certain items that they are selling because customers could acquire these items from other channels of retail. Moreover, an Amazon customer who purchased a perishable physical item may wish to buy this item again, while Warzone players will never purchase virtual in-game repeatedly since digital goods are persistent. As a result, the Warzone store is potentially capable of more accurate personalized choice modeling. For example, the probability of a customer choosing a virtual item they already own will always equal zero. In addition, customers' inventory information may reveal patterns in their preferences. For example, customers who own a particular virtual weapon in-game are more likely to choose cosmetic items for this weapon. In this work, we explore the possibility of incorporating inventory information in choice models, and study the extent to which inventory information can be useful for customer choice modeling.

We propose several *inventory aware* choice models based on random utility. First, we evaluate a classical MNL model and adjust attractions of all items already owned by customers to zero in model estimation. To account for customer heterogeneity, we next incorporate customer segmentation using a clustering algorithm based on purchase history and inventory. We also investigate the effect of owning different items on customers' choices by introducing an *attribute*-based utility structure, where the attributes of each customer are simply his/her items in inventory. This can be viewed as a special case of the *contextual*-MNL model (Chen et al. 2020, Goyal and Perivier 2021). However the large number of virtual items can make this approach intractable for practical use. To address this issue, we further propose a dimension reduction technique based on item grouping. This technique is motivated by modern machine-learning based recommender systems that give a *similarity score* for any pair of items (see, e.g., Von Luxburg 2007); we adopt this similarity score as a distance measure and perform item clustering, reducing the attribute dimension from the number of items to the number of item clusters. The combination of these techniques allows us to achieve substantial performance improvements relative to the standard MNL model.

Our work is related to the machine learning based recommender systems in e-commerce retailers. Recommender systems are a class of information filtering systems that seek to predict the “rating” or “preference” a user would give to an item, with numerous applications such as content recommendation for social media and product recommendation in online retail; we refer to the survey article of Bobadilla et al. (2013) for a comprehensive review. Recommender systems have seen widespread adoption in digital platforms such as YouTube, Netflix and Alibaba. Recently, a field experiment by Feldman et al. (2022) shows that even though machine learning based recommender systems are often superior to discrete choice models in terms of predictive power, they can be inferior in terms of revenue performance. In this work, we combine the strengths of both discrete choice models and recommender systems in that we use item similarity scores of trained recommender systems to group similar items which reduces attribute dimensionality in the MNL model. In our numerical experiments, we compare our inventory aware choice models with a recommender system and show that our approaches yield better revenue performance.

To evaluate the performance of the choice models we propose, we follow the framework in Berbeglia et al. (2021). Specifically, we construct *ground truths* about customers’ choice preferences, and generate in-sample transactional data for choice model estimation. Subsequently, we test the predictive power as well as revenue performance of trained choice models on out-of-sample transactions, which are also generated from the ground truths. We estimate the inventory-aware choice models using maximum likelihood estimation (MLE), and evaluate out-of-sample accuracy using a hard root mean square error (RMSE) metric; we will provide additional details regarding these approaches in Section 3.2. We also evaluate the revenue performance of trained choice models by calculating their respective optimal assortments using the `StaticMNL` algorithm in Rusmevichientong et al. (2010), recording purchases after offering these assortments to customers.

The remainder of this chapter is organized as follows. In Section 3.2, we review some basic discrete choice theory regarding estimation, evaluation and assortment optimization, and describe the concept and structure of ground truths in this study. In Section 3.3, we present our inventory-aware choice models. Section 3.4 evaluates the performance of our inventory-aware choice models via an extensive numerical study, while Section 3.5 concludes.

## 3.2. Estimation and Evaluation of Discrete Choice Models

In this section, we review some basic concepts related to discrete choice theory, including estimation, evaluation, and optimization. Subsequently, we also describe the concept and structure of ground truths used to evaluate our models.

### 3.2.1 Utility Based Discrete Choice Models

Let $\mathcal{N} = \{1, ..., N\}$ denote a universe of substitutable products, and let 0 denote the outside or no-purchase option. A *discrete choice model* describes the customers' choices among this set of alternatives. Formally, consider a situation where a set $\mathcal{S} \subseteq \mathcal{N}$ is offered to the customer, a discrete choice model is a function $P : \mathcal{N} \cup \{0\} \to [0, 1]$ such that $P(j, \mathcal{S})$ denotes the probability that a customer would choose alternative $j$ from $\mathcal{S}$.

A key concept in discrete choice theory is *random utility.* Under random utility models, the customer would obtain a certain level of utility by choosing each of the alternatives in $\mathcal{N} \cup \{0\}$, which we denote $U_0, ..., U_N$. The customer utility for product $j$ can typically be expressed as $U_j = v_j + \varepsilon_j$, where $v_j \in \mathbb{R}$ represents the intrinsic value of product $j$, and $\varepsilon_j$ is a random variable which captures factors that effect utility but are unobservable to the seller. We use the vector $\mathbf{v} = (v_1, ..., v_N)$ to represent product values. Each customer in the population knows his/her own utility, which is a random draw of the random utility vector.

Random utility based choice models rely on two key assumptions. First, these models assume that uncertainty in the utility is due to some some aspect of an individual customer's decision process, which is known to the individual but not to the seller. Second, these models assume that the customer is a rational utility-maximizer. With these assumptions, one can express the choice probability of a customer based on the joint distribution of the random vector $\boldsymbol{\varepsilon} = (\varepsilon_0, ..., \varepsilon_N)$. The most widely used choice model is the multinomial logit (MNL) model, which assumes that the random vector $\boldsymbol{\varepsilon}$ follows independent and identical extreme value type I distribution (i.e., a Gumbel distribution) with mean zero and scale $\beta$. In the MNL model the choice probability can be expressed in closed form

$$P(j, \mathcal{S}) = \frac{e^{\beta v_j}}{\sum_{i \in \mathcal{S} \cup \{0\}} e^{\beta v_i}}, \quad \forall j \in \mathcal{S} \cup \{0\}, \tag{3.1}$$

where $\beta$ is the scale parameter of the Gumbel distribution. In standard MNL, a fundamental property of the choice probability in equation (3.1) is that only differences in utilities of different options matter (see, e.g., Train 2003). To account for this, it is customary to normalize the mean utility without loss of generality (i.e., $v_0 = 0, \beta = 1$).

One major criticism of the MNL is the associated *independence of irrelevant alternatives* (IIA) property, which states that the ratio of the choice probabilities for any two choices remains constant regardless of the other available alternatives in the choice set. The IIA property often empirically fails to hold in real applications, see Ben-Akiva et al. (1985). Many attempts to alleviate this issue have been made, including the mixed or latent-class logit model and the nested logit model. The mixed logit (MMNL) model is obtained by taking a mixture of MNL models, and McFadden and Train (2000) show that any random utility model can be approximated using a mixed logit model by appropriately defining the mixing distribution. The nested logit (NL) model is another commonly used extension to the MNL model. Under the NL model, the universe of products is partitioned into a finite number of subsets (nests); the customer chooses a nest first, and then chooses an alternative within this nest conditioned on the choice of nest. There are also other extensions of MNL, including multi-level nested logit (Kök and Xu 2011) and paired combinatorial logit (Zhang et al. 2020). Overall, the MNL model and its extensions are widely used in pricing and assortment planning due to their interpretability and simplicity.

In addition to the MNL and extensions, there are other utility based choice models that assume a different underlying distribution for the random utilities. The exponomial model (Alptekinoğlu and Semple 2016) assumes that the negative of random utility $\boldsymbol{\varepsilon}$ follows an exponential distribution, while the probit model (Aldrich and Nelson 1984) assumes that $\boldsymbol{\varepsilon}$ follows a joint normal distribution with mean zero and arbitrary variance-covariance structure. In location choice models, $\boldsymbol{\varepsilon}$ enters the utility function through some explicit distance function (Lancaster 1966).

The inventory aware choice models we propose can be viewed as extensions to the MNL model. These extensions are based on the unique characteristics of marketplaces we consider. In particular, we acknowledge customer heterogeneity and perform customer segmentation based on inventory information, which assumes that customers within the same segment share the same choice preferences. We also adjust $\varepsilon_j$ to $-M$ for a sufficiently large $M$ for all products $j \in \mathcal{S}$ already owned by a customer, and account for this adjustment in model estimation. For each customer segment, we further assume that the intrinsic value $v_j$ of product $j$ is a linear function of customer inventories, and propose a novel dimension reduction technique to achieve tractability.

### 3.2.2 Maximum Likelihood Estimation

Estimation is the problem of finding model descriptors that best explain a given set of observed data. We focus on the maximum likelihood estimation (MLE) method, which is among the most widely adopted methods in both academic research and industry practice.

We begin by introducing the data used for estimation. Let $\mathcal{U} = \{1, ..., U\}$ be the customer population. The *training* or *in-sample* data set consists of a sequence of $T$ observations: $\mathcal{T} = \{(u_1, j_1, \mathcal{S}_1, \mathcal{I}_1), ..., (u_T, j_T, \mathcal{S}_T, \mathcal{I}_T)\}$, where $u_t \in \mathcal{U}$ is the customer of transaction $t$, $\mathcal{S}_t \subseteq \mathcal{N}$ is the set of products offered in transaction $t$, $j_t \in \mathcal{S}_t \cup \{0\}$ is the alternative of choice in transaction $t$, and $\mathcal{I}_t \subseteq \mathcal{N}$ is the inventory of customer $u_t$ at the time of transaction $t$. Here, we assume that $j_t \notin \mathcal{I}_t$. Maximum likelihood estimators are based on finding the parameters that maximize the probability of observing the sample data, where likelihood is defined as the probability of the full set of observations occurring independently. Under MLE, given a choice model $\mathcal{M}$ with parameters $\boldsymbol{\theta}$ (which can be multidimensional), the likelihood of observing the set of independent and identically distributed observations equals

$$\mathcal{L} = \prod_{t=1}^{T} P_{\mathcal{M}}(j_t | \mathcal{S}_t, \boldsymbol{\theta}).$$

The MLE problem is to find a vector $\boldsymbol{\theta}$ that maximizes the likelihood function. In practice, the

MLE problem is often solved by maximizing the log-likelihood function, that is, by solving

$$\max_{\boldsymbol{\theta}} \sum_{t=1}^{T} \log P_{\mathcal{M}}(j_t|\mathcal{S}_t, \boldsymbol{\theta}). \tag{3.2}$$

In some special cases, problem (3.2) can be solved in closed form. Otherwise, gradient-based iterative algorithms can be used if $P_{\mathcal{M}}(j_t|\mathcal{S}_t, \boldsymbol{\theta})$ is differentiable.

To illustrate this approach, we demonstrate how the log-likelihood function is constructed for a basic MNL model. In standard MNL, we assume no customer heterogeneity, and the parameters to be estimated are mean utilities of products. The only modification that we make to the standard MNL is that for any $i \in \mathcal{I}_t$, we change the utility of $i$ such that $e^{v_i} = 0$. It follows that the log of the choice probability can be expressed as

$$\log P_t(j_t, \mathcal{S}_t, \mathcal{I}_t) = \log(e^{v_{j_t}}) - \log\left(1 + \sum_{i \in \mathcal{S}_t \setminus \mathcal{I}_t} e^{v_i}\right).$$

We then have the following log-likelihood function

$$\mathcal{LL}(\mathbf{v}|\mathcal{T}) = \sum_{t=1}^{T} v_{j_t} - \sum_{t=1}^{T} \log\left(1 + \sum_{i \in \mathcal{S}_t \setminus \mathcal{I}_t} e^{v_i}\right),$$

which is a real-valued function of the mean utility vector $\mathbf{v}$. The log-likelihood function $\mathcal{LL}(\mathbf{v}|\mathcal{T})$ is differentiable and can be optimized using an off-the-shelf optimizer, such as popular implementations of the BFGS algorithm. For a discussion of MNL estimation via the BFGS algorithm, see Croissant et al. (2012). After the MNL model is trained using in-sample data, we obtain a model that can be used to predict customer choices. In turn, this model can be used in assortment planning or pricing problems. Specifically, let $\hat{\mathbf{v}}$ be the estimated utility vector, that is,

$$\hat{\mathbf{v}} = \arg\max_{\mathbf{v}} \mathcal{LL}(\mathbf{v}|\mathcal{T}).$$

Then, for any out-of-sample transaction with $(\mathcal{S}', \mathcal{I}')$, the choice probability is estimated as

$$P_{\hat{\mathbf{v}}}(j, \mathcal{S}', \mathcal{I}') = \begin{cases} \frac{e^{\hat{v}_j}}{1+\sum_{i \in \mathcal{S}' \setminus \mathcal{I}'} e^{\hat{v}_i}}, & j \in \mathcal{S}' \setminus \mathcal{I}'; \\ 0, & \text{otherwise.} \end{cases}$$

In addition, we also gain managerial insights about the mean products utilities. We implement and test this basic MNL choice model as a benchmark in our later numerical study. Under alternative assumptions on the structure of mean utilities, modifications of $\mathcal{LL}(\mathbf{v}|\mathcal{T})$ need to be made. However, the overall estimation process will remain similar to the steps outlined above.

To conclude, we mention some desirable statistical properties of the MLE method. In cases where the underlying demand model is identifiable and some other mild conditions hold, maximum-likelihood estimators are consistent, asymptotically normal, and asymptotically efficient; we refer to Greene (2003) for a more detailed discussion. In practice, the MLE method is commonly used to estimate MNL choice models and extensions, see, e.g., Berbeglia et al. (2021), Feng et al. (2022).

### 3.2.3 Performance Metrics

We evaluate the performance of our choice models both in terms of predictive power and revenue. To evaluate predictive power, we consider a version of the root mean square error (RMSE). The RMSE is a frequently used measure of the differences between values predicted by a model and values actually observed, providing a sense of how far from the actual ground-truth model the estimated model is. The RMSE is a commonly measure of accuracy for choice-based models in the literature.

To describe the RMSE metric, let $P_{\hat{\boldsymbol{\theta}}}$ be a trained model, and let $\mathcal{T}_{out}$ be the set of out-of-sample transactions. Then, the RMSE can be calculated as

$$RMSE(\mathcal{T}_{out}, P_{\hat{\boldsymbol{\theta}}}) = \sqrt{\frac{\sum_{t=1}^{T} \sum_{j \in \mathcal{S}_t \cup \{0\}} (\mathbb{1}(j_t = j) - P_{\hat{\boldsymbol{\theta}}}(j|\mathcal{S}_t))^2}{\sum_{t=1}^{T} (|\mathcal{S}_t| + 1)}},$$

which is often referred to as the *hard* RMSE.

After a MNL class choice model is estimated, an optimal product recommendation can be obtained for every customer using *assortment optimization*. It is well-documented that models with better predictive power may not always be better in terms of revenue performance. For example, Feldman et al. (2022) show that the MNL model can lead to higher revenues in a field experiment, even though it lacks predictive accuracy when compared to ML-based models. This observation can be explained by the fact that different choice models require different assortment optimization procedures, and it can be challenging to design assortment optimization procedures for choice models that achieve higher predictive accuracy. We emphasize that the computational efficiency of both estimating the choice model using in-sample data and assortment optimization for trained choice models are key concerns in practice.

### 3.2.4 Assortment Optimization

In this section, we briefly discuss how to optimize assortment recommendations given trained choice models. In retail operations, the seller must decide the assortment of products to offer to customers that maximizes revenue given limited display capacity. This problem is called assortment optimization, and is one of the most ubiquitous applications of discrete choice modeling. Formally, let $\mathcal{M}$ be a choice model that predicts choice probabilities for any set of products $\mathcal{S}$. For every product $j \in \mathcal{N}$, we assume a positive revenue $r_j$. To maximize revenue, the seller solves the following problem

$$\mathcal{S}^* = \underset{\mathcal{S}\subseteq\mathcal{N}, |\mathcal{S}|\leq K}{\arg\max} \sum_{j\in\mathcal{S}} r_j P_{\mathcal{M}}(j, \mathcal{S}), \tag{3.3}$$

where a positive integer $K$ is the display capacity. The seller then proceeds to show the customer assortment $\mathcal{S}^*$ to achieve maximum revenue.

In the past two decades, academic research that focuses on efficient algorithm design for assortment optimization under various choice models has expanded rapidly. Talluri and Van Ryzin (2004) show that if customers' choices follow the standard MNL, a revenue-ordered assortments strategy is optimal in the absence of a capacity constraint ($K = |\mathcal{N}|$). Specifically, there exists a revenue threshold, such that simply selecting all products whose unit revenue exceeds this threshold leads to the optimal assortment. Clearly, this strategy can be implemented to be polynomial in time. In the presence of capacity constraints, Rusmevichientong et al. (2010) propose the `StaticMNL` algorithm, which calculates an optimal assortment in polynomial time under the MNL model. Rusmevichientong et al. (2014) consider assortment optimization under the MMNL model, and show that the problem is NP-complete even without capacity constraints. Davis et al. (2014) and

Gallego and Topaloglu (2014) study assortment optimization under the NL model. They establish conditions under which the problem is polynomially solvable, and propose approximation algorithms when these conditions are not met.

The inventory aware choice models that we propose generate a personalized utility vector $\mathbf{v}$ for each customer based on his/her inventory information. Therefore, the assortment optimization problem we face is the capacitated assortment optimization problem under MNL, and we adopt the `StaticMNL` from Rusmevichientong et al. (2010) for assortment planning.

### 3.2.5 Ground Truths

In this section we describe the data generation process we use. As a first step, we create ground truths about customers' choice preferences. Our data generation closely follows Berbeglia et al. (2021). Each instance includes $k = 10$ or $k = 100$ distinctive preferences. We choose to use *ranked lists* to represent customer preferences (also called *stochastic preferences*). Using ranked lists has at least two benefits. First, it has been shown that the stochastic preference model is indeed equivalent to the general random utility class introduced earlier, subsuming all the choice models we consider. Second, the model has a realistic and simple interpretation. Consumers can be clustered into $k$ unobservable types, and each type has a strict preference among the alternatives.

To force some consumers to select the no-purchase option in every assortment, one of the $k$ preference lists contains the no-purchase option as the first alternative for every instance. The probability that a customer is assigned this list equals $p_0 \in \{0.8, 0.2\}$. Each of the remaining ranked lists is selected uniformly and at random out of all the possible permutations. In addition, we generate a positive number between 0 and 1 to represent the weight of each ranked list in the customer population. According to these weights, each customer is assigned a "type" and has a specific preference list. The choices of every customer follow their associated preference list. That is, when offered an assortment the customer iteratively goes through its list: if the alternative is in the offered set but not in his/her inventory, he/she chooses the alternative, and the process concludes with the chosen product added to his/her inventory.

After a ground truth is constructed, the second stage of the data generation process consists of generating in-sample transactional data. At the start of this process, each customer has an empty inventory. The data consists of $T$ time periods, and for every customer, a random assortment set is offered in each period. The customer reacts by following his/her associated preference list, and if a product is chosen it is added to his/her inventory.

After in-sample transactional data is generated, various discrete choice models can be estimated; in addition, similarity-based machine learning recommender systems can also be trained. In our numerical study, we implement a basic machine learning system and investigate its performance relative to our choice model based methods.

To evaluate the predictive power and revenue performance of different approaches, we generate out-of-sample transactions in a similar manner following the ground truths of preferences.

## 3.3. Inventory Aware Multinomial Logit Models

In this section, we describe the inventory-aware choice models that we propose. The baseline approach that we consider is the MNL model outlined in the previous section, which only adjust utilities of products owned by customers compared to standard MNL.

### 3.3.1 Customer Segmentation

Customer segmentation, or *market segmentation*, is the practice of dividing a broad customer market into non-overlapping sub-groups of customers (segments) based on some shared characteristics. In the context of choice models, customers in the same segment share similar product preferences. Customer segmentation has enjoyed a long and fruitful history of academic research in marketing, and we refer to Wedel and Kamakura (2012) for a comprehensive review.

In assortment optimization, identifying customer demand heterogeneity and creating personalized recommendations has received growing attention in recent years and customer segmentation has been considered in Bernstein et al. (2015, 2019), Kallus and Udell (2020). In Bernstein et al. (2015), customer segments are assumed to be readily available and the authors develop a dynamic assortment planning policy. Kallus and Udell (2020) consider a fixed and known but large number of segments; Bernstein et al. (2019) propose a dynamic clustering algorithm which adaptively clusters customers and estimate choice model parameters for each cluster. However, the demographic information they use for customer clustering is low-dimensional.

In our models, we create customer segments based on transactional and inventory information of each customer using the *k-means* algorithm. The k-means algorithm is one of the most well-known and successful unsupervised learning algorithms, and we refer to Hastie et al. (2009) for a detailed treatment of the method. In marketing, the k-means algorithm has been applied in numerous research papers for market segmentation, see, e.g., Kuo et al. (2002), Chiu et al. (2009), Kansal et al. (2018). Recently, Jagabathula et al. (2018) propose a model-based embedding technique for market segmentation with large and unstructured data, which creates a low-dimensional representation vector for each customer used as input for a k-means algorithm.

We briefly review the k-means algorithm and describe how transactional and inventory data is used for model training. Let $\mathcal{U} = \{1, ..., U\}$ be the customer population, and $\mathcal{T} = \{(u_1, j_1, \mathcal{S}_1, \mathcal{I}_1), ..., (u_T, j_T, \mathcal{S}_T, \mathcal{I}_T)\}$ is the in-sample transactional data. Let $\mathcal{T}_u$ denote the subset of $\mathcal{T}$ which include all transactional data of user $u$ for all $u \in \mathcal{U}$. We proceed by creating a vector $\mathbf{d}_u$ from $\mathcal{T}_u$ for all $u \in \mathcal{U}$, which represents relevant information regarding the transactional history and inventory information of customer $u$, which we use as input data of the k-means algorithm. Specifically, for every $u \in \mathcal{U}$, $\mathbf{d}_u$ is a $N$-dimensional vector, such that $d_{u,n} = 1$ if item $n$ is in customer $u$'s inventory and $d_{u,n} = 0$ if item $n$ is offered to $u$ at least one time but the customer never purchased. Otherwise, item $n$ has not been offered to $u$, and the value of $d_{u,n}$ is set to 0.5. Our construction of the training data reflects customers' preferences towards products, and can be viewed as a naive embedding that maps high-dimensional $\mathcal{T}_u$ to the $N$-dimensional space.

Given training data $\mathcal{D} = \{\mathbf{d}_1, ..., \mathbf{d}_U\}$, the k-means algorithm partitions customers into $k$ disjoint clusters, where $k$ is a pre-specified number. Each cluster has a center called *centroid*, and the k-means algorithms aims to iteratively choose centroids that minimize a within-cluster total distance measure, where Euclidean distance is commonly used. For detailed steps of the iterative algorithm, we again refer to Hastie et al. (2009). In this work, we use a popular `scikit-learn` implementation of the k-means algorithm, see Pedregosa et al. (2011a). After the k-means algorithm is trained, we obtain non-overlapping segments of customers, and for each segment, we assume homogeneous product preferences, and proceed to estimate choice model parameters using transactional data containing only users in the segment.

### 3.3.2 Contextual Multinomial Logit

In addition to using transactional and inventory information for customer segmentation, we also explore the possibility of improving MNL estimation by using it as contextual information. This is motivated by the fact that ownership of certain products may alter customers' preferences over other products. Consider for example a player in Call of Duty: Warzone and suppose he/she owns a certain weapon. Then, attachments that make this weapon more powerful in combat will likely be more attractive than attachments of another weapon not in his/her inventory.

Formally, for a customer in a generic segment, let $\mathcal{I}$ be the set of products that he/she owns. Then, we assume that the utility of item $j \in \mathcal{N} \setminus \mathcal{I}$ is an affine function of $\mathcal{I}_u$, that is,

$$u_j = v_{j,0} + \sum_{i \in \mathcal{I}} v_{j,i},$$

where $v_{j,0}$ is the base utility of product $j$, and $v_{j,i}$ accounts for the impact of owning product $i$ on the customer's utility of product $j$. With this assumption, we can express the log-likelihood function as

$$\mathcal{LL}(\mathbf{v}|\mathcal{T}) = \sum_{t=1}^{T} \left( v_{j,0} + \sum_{i \in \mathcal{I}_t} v_{j,i} \right) - \sum_{t=1}^{T} \log \left( 1 + \sum_{i \in \mathcal{S}_t \setminus \mathcal{I}_t} e^{(v_{i,0} + \sum_{k \in \mathcal{I}_t} v_{i,k})} \right). \tag{3.4}$$

While the baseline MNL model has $N$ parameters, equation (3.4) has $N \cdot (N+1)$ parameters. When the number of products is large, this can lead to computational inefficiencies.

The above model can be viewed as a special case of the *contextual MNL* model, where product utilities are functions of certain customer attribute information. The contextual MNL model has been investigated in assortment optimization and pricing recently by Wang et al. (2019), Chen et al. (2022), among others. The log-likelihood function (3.4) can be constructed for each customer segment, and optimized using an off-the-shelf optimizer; when training is complete, a customer's choice probabilities are obtained by simply plugging in the utilities based on his/her inventory.

Because the number of parameters in contextual MNL is often large, it is prone to overfitting. To mitigate overfitting risks, we implement regularization in our estimation. In particular, we add $L2$ regularization term $-\lambda \sum_{i=1}^{N} \sum_{1}^{K} v_{i,k}^2$ to the log-likelihood function in equation (3.4), where $\lambda$ is a positive regularization parameter. Alternatively, one could use other regularization techniques such as the $L1$ norm or the elastic net.

### 3.3.3 Dimension Reduction with Item Clustering

As mentioned in the previous section, using inventory information for the contextual MNL model can make it prohibitively expensive to train the model. Such dimensionality issue in contextual MNL models are recognized in Wang et al. (2019), who propose to reduce the dimensionality by combining LASSO regularization and random projection.

To alleviate these computational issues, we propose a dimension reduction technique based on item clustering. The idea behind our approach is to divide items into disjoint clusters: instead of keeping track of the impact of ownership for every item pair, we count the number of items owned by each customer in every item cluster, reducing the number of parameters. Our approach is related to Miao et al. (2019), where the authors perform clustering of products with similar demand patterns to pool data of low-sale products for estimation. Before we describe how item clustering is achieved, we demonstrate how the number of dimensions is reduced given item clusters.

Suppose items are divided into $C$ disjoint clusters, that is, $\mathcal{N} = \mathcal{N}_1 \cup \mathcal{N}_2, ..., \cup \mathcal{N}_C$, and $\mathcal{N}_i \cap \mathcal{N}_J = \emptyset$ for all $i = 1, ..., C, j = 1, ..., C, i \neq j$. As a first step we create an attribute vector $\mathbf{g} = \{g_1, ..., g_C\}$ for a customer in a generic customer segment, where $g_c$ is the number of items in cluster $c$ that the customer owns at the time of the transaction. We assume that the customer's utilities depend on $\mathbf{g}$, i.e.,

$$v_j = v_{j,0} + \sum_{c=1}^{C} g_c v_{j,c},$$

for all $j \in \mathcal{N} \setminus \mathcal{I}$. The corresponding log-likelihood function is then

$$\mathcal{LL}(\mathbf{v}|\mathcal{T}) = \sum_{t=1}^{T} \left( v_{j,0} + \sum_{i=c}^{C} g_{t,c} v_{j,c} \right) - \sum_{t=1}^{T} \log \left( 1 + \sum_{i \in \mathcal{S}_t \setminus \mathcal{I}_t} e^{(v_{i,0} + \sum_{c=1}^{C} g_{t,c} v_{i,c})} \right). \tag{3.5}$$

The number of parameters that require estimation is reduced from $N \cdot (N+1)$ to $N \cdot (C+1)$. Similar to contextual MNL, we also employ $L2$ regularization when estimating the parameters. In the numerical study, we demonstrate the effectiveness of our approach.

The item clustering method that we employ is the *spectral clustering* algorithm. At a high level, spectral clustering extracts the most essential information in the features and projects it onto a low-dimensional space where it is easier to identify the underlying cluster structure. The application of spectral clustering to demand estimation was investigated in Keskin et al. (2022), where the authors use spectral clustering for customer segmentation.

Given a set of data points and some notion of similarity between all pairs of data points, the goal of clustering is to divide the data points into several groups such that points in the same group are similar and points in different groups are dissimilar to each other. One way of representing the data is in the form of a similarity graph, where each vertex represents one data point, and weighted edges represent a similarity measure between vertices. Partitioning vertices in the similarity graph into subsets such that the total weight of edges connecting subsets is minimum and sizes of subsets are similar is NP-hard, and spectral clustering presents an approximate solution to this problem. For a detailed analysis and discussion of the algorithm, we refer to Von Luxburg (2007).

A notion of similarity between items is required to implement the spectral clustering algorithm. Motivated by research on recommender systems we adopt a collaborative filtering approach to obtain item similarities from inventory information. We describe this approach in the next section.

### 3.3.4 Collaborative Filtering and Item Similarity

The use of recommender systems has grown significantly in recent years, partly due to the explosion of internet-based commerce and content consumption. A common task of recommender systems is to improve customer experience through personalized recommendations based on prior implicit feedback. One of the most widely adopted strategies is known as *collaborative filtering* (CF), which analyzes relationships between users and interdependencies among products to identify new user-item associations. For example, certain CF systems identify pairs of items that tend to be rated similarly or pairs of users with a similar history of rating or purchasing to deduce unknown relationships between users and items. The only required information is the past behavior of users, e.g., previous transactions.

The application of CF systems has been successful in various domains, including content recommendation in Netflix (Bennett et al. 2007), item recommendation on Amazon (Linden et al. 2003), and personalized recommendation in Call of Duty: Warzone. Instead of applying the CF method directly, we train the CF algorithm using in-sample transactions to obtain item-to-item similarities, and subsequently use the similarities as inputs to the spectral clustering algorithm for item clustering. We use the algorithm proposed in Hu et al. (2008), who develop a system that uses only *implicit* feedback data. Specifically, purchase records can be viewed as positive feedback from customers, though substantial evidence of customers' negative feedback is lacking. In our setting, the implicit CF model is trained using the in-sample transactional data, after which similarity between any pair of items can be obtained.

### 3.3.5 Summary of Approach

Combining the techniques introduced earlier in this section, the inventory-aware MNL model that we develop can be outlined as follows. First, we conduct customer segmentation using the k-means algorithm to create disjoint customer segments. Second, for each customer segment we obtain item similarity by training an implicit feed CF model. Using the item similarities from the CF model as input, we create item clusters using the spectral clustering algorithm. Third, the segment-wise contextual MNL model is estimated by optimizing the log-likelihood function in equation (3.5). After the inventory-aware MNL model is trained, customer-specific utility vector are obtained based on each customer's segment and inventory information, and personalized assortment recommendations are determined using the assortment optimization model.

We conclude the section by noting that certain components of the inventory aware choice models can be replaced by alternative methods, including the customer segmentation algorithm, item similarity scores, item clustering algorithm and regularization techniques. Careful investigation and evaluation of alternative approaches and fine-tuning model parameters could lead to further improvements in performance.

## 3.4. Numerical Experiments

To assess the performance of different choice models, we create randomly generated instances and conduct a numerical study. In this section, we first introduce the approaches we evaluate in our experiments. Subsequently, we describe instance generation details and present our results and findings.

### 3.4.1 Experimental Setup

We compare four approaches on a wide range of instances. All experiments are implemented in `Python 3.8` on a PC running Windows 10 with a Intel Core i5-12600k (10 cores@4.6 GHz) and 32 Gb of RAM. For MLE estimations, we use the `L-BFGS-B` algorithm provided by the `scipy` (Virtanen et al. 2020) package. The k-means algorithm for customer segmentation as well as the spectral clustering algorithm for item grouping utilize the `scikit-learn` package (Pedregosa et al. 2011b). The collaborative filtering approach with implicit feedback uses `implicit` package (Frederickson 2022). When using the above open source packages, we try to use them in an off-the-shelf state with minimum tuning. When testing revenue performance of utility based choice models, we implement and use the `StaticMNL` algorithm in Rusmevichientong et al. (2010).

**MNL.** The first approach that we evaluate is the basic multinomial logit model described in Section 3.2.2. As mentioned, the only difference of our implementation compared to the standard MNL model is that we explicitly set utilities of items owned by players to 0. Parameters to be estimated of this approach are the intrinsic utilities of items, and we estimate these parameters by optimizing the log-likelihood function directly.

**MNL-Seg.** The next approach that we evaluate is the multinomial logit model with customer segmentation. We try to capture customer preference heterogeneity by learning from historical activities and group customers with similar tastes together; details of this approach is described in Section 3.3.1. In our numerical study, we cluster customers into 10 disjoint segments. Subsequently, we estimate segment-wise utilities by building and optimizing log-likelihood functions for each segment.

**CF.** We also evaluate a collaborative filtering algorithm in the numerical study. The purpose of evaluating collaborative filtering algorithms is to investigate the performance of popular machine learning based recommender systems relative to utility based choice models. In our experiments, the CF approach lacks the ability to predict personalized choice probabilities given different assortments, but does make personalized recommendations based on item and customer similarities. As a result, we only test its revenue performance and not predictive power. Further details for this approach can be found in Section 3.3.4.

**IA-MNL.** The last approach that we test is the "inventory aware MNL" model that combines all the components described in the previous section. In particular, this approach involves customer segmentation and item clustering based contextual MNL. In our experiments, we use 10 customer segments and 5 item clusters. It is worth noting that we also implemented the contextual MNL approach with dimension reduction, but found that the training was overly inefficient in our experiments.

### 3.4.2 Description of Instances

To create instances we closely follow the approach outlined in Berbeglia et al. (2021). We generate instances across a variety of different settings and try to be comprehensive, even though the instances that we evaluate are significantly larger that those in Berbeglia et al. (2021). For each instance specification, we generation 10 instances and track the average performance of the approaches we evaluate.

In the first stage of instance generation we create preference lists. We assume that there are 50 candidate items in the store (each with a random unit price uniformly drawn from $[0, 1]$) and a population of 1000 customers. Each customer is associated with either $k = 10$ or $k = 100$ distinct preference lists. Among all customers, we force certain customers to always choose the no-purchase option with probability $p_0 \in \{0.2, 0.8\}$.

The second stage of instance generation consists of creating in-sample transactions. We assume a *cold start* phase of $T = 20$ or $T = 40$ periods. At the start, every customer has an empty inventory. In each period, every customer is shown a random assortment of size $S \in \{3, 5\}$ uniformly drawn from the set of items. Each customer then proceeds to choose by following his/her associated preference lists, and their inventory is updated accordingly. Table 3.1 illustrates the in-sample transaction records that we generation for an instance.

Table 3.1: Sample in-sample transactional data

| customer id | inventory | assortment | choice |
|---|---|---|---|
| 1 | [ ] | [0, 2, 3, 5] | 2 |
| 1 | [2] | [0, 4, 7, 8] | 7 |
| 1 | [2, 7] | [0, 5, 7, 9] | 0 |
| 1 | [2, 7] | [0, 3, 6, 9] | 3 |
| 1 | [2, 3, 7] | [0, 4, 5, 9] | 0 |

After in-sample transaction data is generated for each instance, we train the four approaches mentioned earlier. To evaluate trained models, we generate random out-of-sample transactions, and and calculate RMSE as a measure of predictive power. To evaluate revenue performance, we make assortment recommendations for each customer based on the choice models and track the customers' choices.

### 3.4.3 Results

To understand how different approaches perform relative to one another under different environments, we create instances with different parameters. The test instances we create vary the latent number of preference lists $k \in \{10, 100\}$, the percentage of no-purchase customers $p_0 \in \{0.2, 0.8\}$, the assortment size $K \in \{3, 5\}$ and the cold start periods $T \in \{20, 40\}$. For each setting, we generate 10 random instances, and track the average RMSE, average revenue and average training time of different approaches.

To facilitate the interpretation of results, we first discuss high level findings before presenting detailed results. First, the CF approach performs worse than all utility based choice models in terms of revenue performance, even though it is very fast to train, On a high level, the MNL-Seg model significantly outperforms the MNL baseline. The IA-MNL model shows further improvements over MNL-Seg, and is particularly useful when the underlying ground truths of customers is complex, and when the training data sets are sufficiently large.

#### 3.4.3.1 Estimation efficiency

In our numerical experiments, all the algorithms implemented by open-source packages (including the CF approach, k-means clustering and spectral clustering) come with sophisticated performance optimization and are very efficient for our instances. Specifically, they all require less than 1 second for all instances.

To compare estimation efficiency of utility-based models, we optimize the log-likelihood functions using the `L-BFGS-B` algorithm implemented by `scipy` without multi-threading. The MNL model is by far the fastest to estimate. The efficiency of the MNL model for our instances is comparable to that of Berbeglia et al. (2021), where the authors report an average time of 4 seconds when estimating their largest instances with 10 items and 6,000 rows of transactional records. In our experiments, it takes 15.6 seconds when estimating instances with 50 items and 20,000 rows

(1,000 customers with 20 cold-start periods), and 32.8 seconds when estimating instances with 50 items and 40,000 rows.

Estimation of the MNL-Seg model requires estimation of a MNL model for each customer-segment. For our instances, estimation of MNL-Seg takes an average of 67.4 seconds for $T = 20$ and 144.7 seconds for $T = 40$. Since the k-means algorithm for customer segmentation uses $k = 10$, estimation of each segment takes 6.7 and 14.5 seconds for $T = 20$ and $T = 40$ respectively, faster than estimation of the basic MNL due to a smaller in-sample data for each segment. With straightforward parallelization, the MNL-Seg model could be as efficient as the MNL model.

Estimation of the IA-MNL model is significantly more demanding, due to a much larger number of parameters. In fact, there are 300 parameters for the IA-MNL model for each customer segment compared to 50 parameters for the MNL-Seg model. In our experiments, estimation of IA-MNL takes 3794 seconds for $T = 20$ and 8845 seconds for $T = 40$. Still, with a parallel implementation and further performance acceleration, the estimation time of IA-MNL could be improved substantially.

We note that there are 2,550 parameters for the contextual-MNL model without dimension reduction by item grouping in contrast to 300 for IA-MNL; when the number of items equals 200, the number of parameters to be estimated is 1,200 for IA-MNL and 40,200 for contextual-MNL.

#### 3.4.3.2 Impact of ground truth complexity

We investigate the impact of ground truth complexity when creating instances, and show results of different models depending on the number of latent customer preferences. We refer to the $k = 10$ case as the *less complex* scenario, and the $k = 100$ case as the *more complex* scenario. Results are reported in Table 3.2.

We view MNL as the baseline approach, and values inside parentheses are performances relative to the MNL approach. Comparing CF with MNL, we find that the basic MNL approach outperforms CF across all instances, and the difference is especially significant for the more complex instances. This is consistent with results in Feldman et al. (2022), where the authors find that basic

Table 3.2: Results of instances with $k = 10$ (left) and $k = 100$ (right)

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.245 | 214.4 |
| MNL-Seg | 0.028 (88.3) | 338.3 (57.7) |
| IA-MNL | 0.030 (87.5) | 338.5 (57.8) |
| CF | — | 201.8 (-5.85) |

(a) Results of instances with $k = 10$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.248 | 212.8 |
| MNL-Seg | 0.202 (18.6) | 245.0 (15.1) |
| IA-MNL | 0.191 (22.9) | 267.2 (25.5) |
| CF | — | 169.7 (-20.2) |

(b) Results of instances with $k = 100$

MNL can lead to higher revenue compared to sophisticated machine learning based recommender systems. When comparing MNL-Seg and IA-MNL with the MNL baseline, we find that both approaches remarkably outperform the MNL baseline in terms of both predictive power and revenue. This confirms the value of the customer segmentation idea.

An important observation is that performance of different models strongly depends on the complexity of ground truths. The performance of all models except the MNL is dramatically better for less complex instances, with more accurate predictions and higher revenue. For less complex instances, the RMSE values of MNL-Seg and IA-MNL are approximately an order of magnitude smaller than the RMSE of MNL, indicating that the k-means customer segmentation algorithm performs really well when underlying ground truths are less complex. In real world applications, however, it is unlikely that there are only 10 distinctive customer preferences, so we expect the results for more complex instances to be more relevant. For more complex instances, the relative improvement of IA-MNL over MNL-Seg is more significant, and IA-MNL generates approximately 9% more revenue than MNL-Seg.

To better isolate and understand the effects of instance settings, we only report results for more complex instances with $k = 100$ in the remainder of this section.

#### 3.4.3.3 Impact of market share

We compare the performance results of approaches under two scenarios: high market share ($p_0 = 0.2$) and low market share ($p_0 = 0.8$). For each instance, $p_0$ refers to the fraction of customers who prefer the no purchase option over all items. We refer to $1 - p_0$ as the market share.

We report average performance of the various approaches in Table 3.3. We start by observing that the relative order of performance (IA-MNL, MNL-Seg, MNL and CF) remains the same as in the previous section. Next, we observe that the RMSE of all methods decreases when $p_0 = 0.8$; this can be explained by the fact that prediction becomes easier as more customers prefer the no purchase option. Of course, the revenue collected for $p_0 = 0.8$ is significantly less than revenue of $p_0 = 0.2$.

Table 3.3: Results of instances with $p_0 = 0.2$ (left) and $p_0 = 0.8$ (right)

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.325 | 340.1 |
| MNL-Seg | 0.265 (18.4) | 394.6 (16.0) |
| IA-MNL | 0.247 (24.0) | 434.3 (27.6) |
| CF | — | 279.6 (-17.7) |

(a) Results of instances with $p_0 = 0.2$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.172 | 85.43 |
| MNL-Seg | 0.139 (18.9) | 95.42 (11.6) |
| IA-MNL | 0.136 (20.8) | 100.1 (17.1) |
| CF | — | 59.79 (-30.0) |

(b) Results of instances with $p_0 = 0.8$

The relative improvement of MNL-Seg over MNL is similar for high market share and low market share, but the improvement of IA-MNL over MNL-Seg is more significant for high market share instances. Specifically, for high market share instances, the RMSE of IA-MNL is 8% better than that of MNL-Seg, and the revenue of IA-MNL is 10% more. Considering the popularity of marketplaces such as multiplayer video games, this difference can be significant.

#### 3.4.3.4 Impact of assortment size

We show results on instances with different assortment sizes in Table 3.4. Assortment capacity is often determined by the design of website interfaces of marketplaces, and we aim to obtain some insights regarding the choice model estimation or assortment optimization aspects. Our main observation is that when assortment size is larger, the benefit of adopting IA-MNL and MNL-Seg over MNL is significantly smaller. It is known that a smaller assortment size enforces customer substitution behavior; with a larger assortment size, substitution behavior is less emphasized, as customers tend to find their most preferred option even only MNL is used. For practitioners, if the assortment capacity is constrained by interface design to be small, adopting more sophisticated MNL-Seg or IA-MNL could be more profitable.

Table 3.4: Results of instances with $K = 3$ (left) and $K = 5$ (right)

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.270 | 208.8 |
| MNL-Seg | 0.215 (20.2) | 258.4 (23.7) |
| IA-MNL | 0.207 (23.2) | 285.5 (36.7) |
| CF | — | 174.0 (-16.6) |

(a) Results of instances with $K = 3$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.227 | 216.8 |
| MNL-Seg | 0.189 (16.7) | 231.6 (6.82) |
| IA-MNL | 0.176 (22.6) | 248.9 (14.8) |
| CF | — | 165.4 (-23.7) |

(b) Results of instances with $K = 5$

#### 3.4.3.5 Impact of volume of training data

We report results with different volume of in-sample training data in Table 3.5. When generating in-sample data, we use a short $T = 20$ cold-start period (smaller data) and a long $T = 40$ cold-start period (larger data). As is to be expected, more data gives models with more parameters strong performances. Specifically, the relative RMSE improvement of IA-MNL over MNL-Seg is 13% for larger training data, and the revenue improvement is 15%. It is interesting to note that for smaller training data, even though the RMSE of IA-MNL is the same as the RMSE of MNL-Seg, and IA-MNL leads to a revenue improvement of 5%. When there is sufficient training data, it could be beneficial to adopt the IA-MNL model, whereas the MNL or MNL-Seg approaches may be suitable when training data is limited.

Table 3.5: Results of instances with $T = 20$ (left) and $T = 40$ (right)

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.284 | 261.9 |
| MNL-Seg | 0.229 (19.2) | 296.9 (13.3) |
| IA-MNL | 0.230 (18.8) | 311.3 (18.8) |
| CF | — | 191.5 (-26.8) |

(a) Results of instances with $T = 20$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.213 | 163.6 |
| MNL-Seg | 0.175 (17.8) | 193.1 (17.9) |
| IA-MNL | 0.152 (28.4) | 223.0 (36.2) |
| CF | — | 147.9 (-9.63) |

(b) Results of instances with $T = 40$

## 3.5. Concluding Remarks

In this work, we study a customer choice modeling problem faced by centralized online marketplaces. Motivated by online multiplayer video games, where the store operator has a complete history of customers' assortments and choices, we set out to explore ways in which this information can be exploited to understand customers' choice behaviours. We propose an inventory aware MNL choice model, where we first conduct customer segmentation to identify heterogeneous customer preferences, and then estimate utilities of items based on customers' ownership of other items. The potential high-dimensionality of items calls for a dimension reduction method to achieve computational tractability. We propose an item clustering approach based on similarity, which is in turn obtained by a collaborative filtering method.

We evaluate the performance of the proposed approach through a numerical study based on randomly generated preferences. We find that utility based choice modeling can result in a substantial improvement in revenue performance compared to collaborative filtering. We also observe that MNL with customer segmentation can lead to substantial improvements in predictive power and revenue compared to a MNL without customer segmentation, while still being efficient to estimate. Finally, our experimental results show that the item clustering based contextual MNL can lead to significantly better performance, although it is markedly harder to estimate.

There are a few directions to pursue future research. First, in our numerical experiments, preferences are generated with no assumptions on structures of preferences. Under situations when there are significant patterns of structures, e.g., substitute items or complementary goods, contextual MNL models could be more powerful in capturing these effects, and more numerical experiments can be designed to understand the relative order of performance of different models in those environments. Although we briefly discuss the potential application of an inventory aware NL model in the appendix, we do not implement and test this model in our experiments. One could also study inventory aware versions of other choice models, including the Markov chain model and the exponomial model, and see if they can be useful under certain operational environments.

# Chapter 4

# Assortment Optimization for Online Multiplayer Video Games

**Abstract.** Problem Definition: We consider an assortment optimization problem for a class of online multiplayer video games, where the in-game virtual store has a unique structure with two sections, *Featured* and *Just For You (JFY)*. All customers (players) are offered the same Featured section assortment whereas the JFY section is used for personalized recommendations. Academic/Practical relevance: Our work is the first to study assortment optimization for the gaming industry under discrete choice models; it is also the first to devise solution approaches for the constrained mixture-of-nested-logit model with performance guarantees. Methodology: We model the choice of customers under the constrained mixture-of-nested-logit model, and design a fully polynomial time approximation scheme (FPTAS), as well as a mixed integer linear programming (MILP) formulation. We also provide a Lagrangian upper bound and a fast heuristic algorithm. Results: We provide theoretical performance guarantees for the FPTAS, the MILP formulation, and the heuristic algorithm. Numerical experiments show that our approaches perform well across a variety of settings. Managerial implications: Our work provides guidance for how online video game stores can manage assortments for effective revenue maximization. We propose an "assortment" of solution approaches that allows practitioners to choose a method that best suits their settings.

## 4.1. Introduction

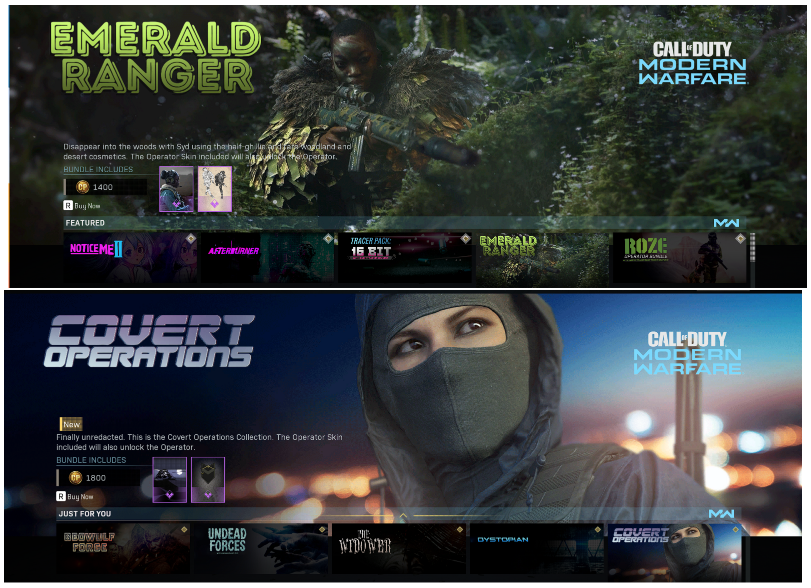
Since the start of the COVID-19 pandemic, online multiplayer video games are reaching more people as a much-needed outlet for connection in isolation. A recent market report by a gaming intelligence company (Newzoo 2020) states that there are over two billion video game players worldwide, and the gaming market will exceed $200 billion by the year of 2023. In recent years, a significant and increasing portion of revenue is generated from so-called microtransactions, where virtual items are sold through micropayments that typically involve small sums of money. In particular, virtual items are displayed in a store interface within the game client/app, and customers spend in-game or real-world currency to purchase these items either to gain a competitive advantage over other players, or for cosmetic purposes. According to Newzoo (2020), over 70% percent of game revenues come from microtransactions, surpassing traditional full game sales. Two of the biggest gaming publishers, Activision Blizzard and Electronic Arts, announced $956 million and $789 million of revenue from microtransactions in the first quarter of 2020, respectively. The rapid growth of such microtransactions has motivated gaming companies to find opportunities that both enhance the customer experience and increase revenue, using the abundance of transactional and contextual data that is available by the very nature of online video games.

When customers of the game visit the virtual store (the seller) that sells in-game items, they are presented with a set of items simultaneously, known as an assortment, to induce them to purchase. Due to the capacity of the store interface, only up to a limited number of items can be displayed, and the seller has to decide which assortment to offer in order to maximize the expected revenue; this is known as assortment optimization (Gallego et al. 2019). Although assortment optimization in traditional online retailing is becoming increasingly pervasive and successful, its adoption by gaming companies is still scarce.

The layout of online multiplayer video game stores often has a common structure, with a *Featured* section intended for promoting new or popular items, and a *Just For You (JFY)* section tailored to each customer's taste. The Featured assortment is offered to all customers whereas the JFY section can be personalized. For example, Figures 4.1 and 4.2 show the store interfaces of two of the most financially successful games, Call of Duty: Warzone and Fortnite. Call of Duty: Warzone is a free-to-play video game published by Activision Blizzard; three months after its release in March 2020, microtransactions in the game generated almost $500 million in revenue.

Fortnite, on the other hand, is a free-to-play game released in 2017 by Epic Games; in 2019, Fortnite brought in revenues of \$1.8 billion, all of which came from microtransactions. The unique structure of online multiplayer video game stores as well as the magnitude of financial implications inspire us to investigate the assortment optimization problems that arise in this setting.

Figure 4.1: Store Interface of Call of Duty: Warzone

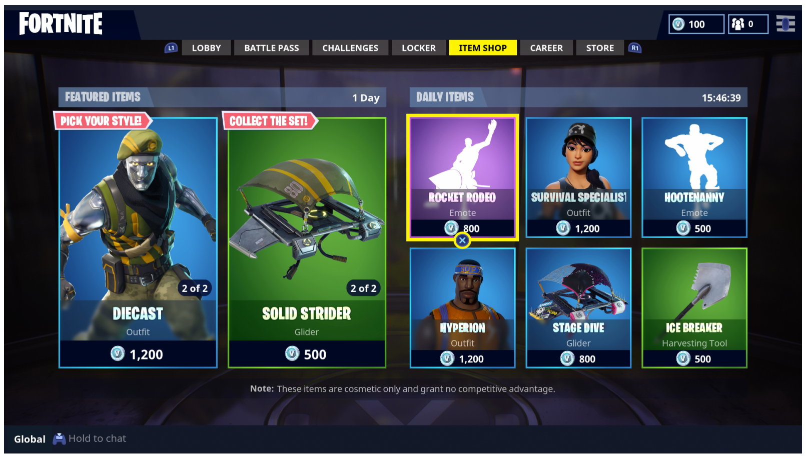


Disappear into the woods with Syd using the half-ghillie and rare woodland and desert cosmetics. The Operator Skin included will also unlock the Operator.

Finally unredacted. This is the Covert Operations Collection. The Operator Skin included will also unlock the Operator.

In online multiplayer video game stores, it is usually the case that customers can only purchase items offered in the assortment; there is no place to go in the store to browse and buy from a complete list of all items, or search for a certain item. The Featured section usually includes highly attractive and popular items, which often come with in-game content updates. A shared, non-personalized Featured section ensures that when customers see these items (e.g., the newest weapon) in game sessions used by friends or opponents and find them attractive, they can go to the store and purchase them. On the other hand, a personalized JFY section can account for heterogeneity of customer preferences, and improve revenue by offering customers items that more accurately match their taste, see Arora et al. (2008); in addition, it attracts customer attention and fosters customer loyalty and lock-in (Ansari and Mela 2003). It is important to note that the Featured and the JFY section assortments usually draw from non-overlapping pools of items.

Figure 4.2: Store Interface of Fortnite



**Note:** These items are cosmetic only and grant no competitive advantage.

Currently, gaming companies mostly use Featured assortments hand picked by developers in an ad-hoc fashion, and machine learning-based methods to determine JFY items. In this manuscript, we propose a modeling framework to systematically select both the Featured and JFY section assortments to maximize expected revenue.

The structure of such stores naturally motivates us to model customers' choices using a nested logit model. As a popular extension to the well known multinomial logit model (MNL) pioneered by Luce (1959) and McFadden et al. (1973), the nested logit model was introduced by Williams (1977). Under the nested logit model, items are organized in nests; the choice process of a customer proceed in a way such that the customer first select a nest or leave without purchase, then select an item in the selected nest. We model the Featured section and the JFY section as two nests, each with a display capacity limited by the store interface.

Our work complements a stream of research of assortment planning under discrete choice models focusing on developing algorithms for the corresponding revenue maximization problems. In particular, we develop solution algorithms for the constrained mixture-of-nested-logit model, where customers following a nested logit model belong to multiple segments, each with different nested logit choice model parameters, and the assortments are subject to display capacity constraints. The constrained-mixture-of-nested-logit model provides flexibility in modeling customers' choices, but is provably difficult to solve. To the best of our knowledge, our work is the first to devise solution algorithms for the constrained mixture-of-nested-logit model with performance guarantees. Although our approach is motivated by online multiplayer video games, our solution algorithms are not limited to the gaming industry and can also be used in other applications.

The use of a JFY assortment is closely related to an emerging stream of research on assortment personalization. This is the practice of offering items tailored to each customer's taste based on previously collected data. Due to a rapid growth of information technology, retailers collect an increasingly large amount of customer data, and personalized assortment planning has received considerable attention in recent years. Much of the research in this area assumes that the seller has limited prior information of customers' preferences, and proposes efficient learning algorithms to address the *exploration* versus *exploitation* trade-off. While the segmentation of customers and estimation of choice model parameters are interesting and important research questions in and of itself, we restrict our attention to solving a static revenue maximization problem. Throughout this chapter we assume that customer segmentation is readily available, and that we know the size and choice model parameters for each segment. We base this assumption on the fact that gaming companies often have substantial resources dedicated to analyzing customers' preferences and conducting behavioral segmentation.

### 4.1.1 Main Contributions

The contributions of this chapter are as follows.

- We formulate the assortment optimization problem for a class of online multiplayer video games with a specific structure, where a Featured section contains items offered to all customers, and where a JFY section allows for personalized item recommendations. We model the choices of customers under a constrained mixture-of-nested-logit model, and show that analyzing the problem through the lens of discrete choice modeling and assortment optimization can lead to competitive revenue performance.

- We propose a fully polynomial time approximation scheme (FPTAS) when the number of customer segments is fixed. The high level idea of the FPTAS is to divide and conquer, that is, creating guesses on relevant function terms and trying to find an feasible assortment for each guess.

- We propose a MILP formulation based on approximating the nonlinear and non-convex objective function of the revenue maximization problem with a piecewise linear function. Our formulation has both a theoretical performance guarantee and strong computational performance.

- We design a heuristic algorithm for finding feasible assortments, and prove an approximation bound for the heuristic. Our numerical study shows that the heuristic can be solved efficiently across all instances and consistently achieves near optimal performance.

- We construct a Lagrangian upper bound by conducting a grid search on certain components of the formulation. The upper bound can be used to evaluate performance of the heuristic algorithm.

- More generally, we highlight the potential benefits of assortment optimization in the rapidly growing gaming industry, where its unique structure and unlimited supply of data provide a number of novel challenges.

### 4.1.2 Related Literature

Our work is closely related to a stream of literature that aims to solve the assortment optimization problem under various discrete choice models. In their seminal work, Talluri and Van Ryzin (2004) study an assortment optimization problem under MNL where the parameters are deterministic and known; they show that the optimal assortment includes a certain number of items with the highest revenues, often referred to as revenue-ordered assortments. Rusmevichientong et al. (2010) study the assortment optimization problem under MNL with a capacity constraint, and propose a polynomial time algorithm. Davis et al. (2014) characterize conditions under which the assortment optimization problem under nested logit is polynomially solvable. Gallego and Topaloglu (2014) study a class of constrained assortment optimization problems under nested logit and propose a polynomial time algorithm when the assortments are cardinality constrained. Rusmevichientong et al. (2014) study the assortment optimization problem under MNL with multiple customer segments, each with different preferences for the items; this choice model is called the mixture-of-logits model. They show that the mixture-of-logits problem is NP-complete, and give performance guarantees for a certain class of assortments. Feldman and Topaloglu (2015a) are the first to study assortment optimization under the mixture-of-nested-logit model, where customers who follow a nested logit model belong to multiple segments, each with different nested logit choice model parameters. They focus on bounding the optimal expected revenue using a Lagrangian relaxation, and use the resulting bounds to evaluate the quality of heuristic solutions. In this chapter, we extend their work and propose approximation algorithms for the mixture-of-nested-logit model with both theoretical guarantees and strong numerical performance.

Our work is also related to assortment personalization. Bernstein et al. (2019) and Kallus and Udell (2020) model the heterogeneity of customers by customer segmentation, and offer personalized assortments for each segment; another stream of research takes a different approach, and models customers' preferences as a function of their attribute vectors; see Cheung and Simchi-Levi (2017), Chen et al. (2021a), Miao and Chao (2019), Wang et al. (2019). While most of the assortment personalization research focuses on dynamically learning and optimizing personalized assortments, we study the static optimization problem assuming segmentation and parameter estimation are known. There are two reasons that drive us toward this approach. First, the static problem is an interesting and challenging problem with practical relevance in and of itself due to the unique structure of the video game stores. In addition, online multiplayer video game stores are usually updated on a daily basis, and millions of store visit records of the previous day make it possible to update estimated parameters every day before the assortment optimization algorithms are run. This setting is somewhat different from traditional online retailers, who may need to come up with personalized assortments for each arriving customer.

Our FPTAS is closely related to research on approximation algorithms for assortment optimization under various choice models. Due to hardness results for variants of the assortment optimization problem, approximation algorithms and approximation schemes provide an attractive alternative to achieve tractability. In Davis et al. (2014), the authors show that the assortment optimization problem under certain types of the nested logit model is NP-hard, and propose approximation algorithms with instance specific guarantees; Rusmevichientong et al. (2014) show that the mixture-of-logit model is NP-complete with only two customer segments, and derive tight approximation results for revenue-ordered assortments; Feldman and Topaloglu (2015b) develop a 4-approximation algorithm for the nested logit model with a capacity constraint across all nests; Aouad et al. (2018) show tight approximation bounds for assortment optimization under a general rank-based choice model and provide approximation algorithms that attain best possible performance; Désir et al. (2020) study a class of capacitated assortment optimization problems and provide a framework to support the design of an FPTAS; Liu et al. (2020) propose an FPTAS for a multistage assortment optimization problem under MNL; Désir et al. (2021) consider assortment optimization under the so-called mixture-of-Mallows model, and propose a PTAS as well as a MIP formulation; Lyu et al. (2021) investigate capacitated assortment optimization under a multivariate MNL model where customers may buy more than one item, and propose FPTASes for several variants of the problem. In this work, we propose a FPTAS when the number of customer segments is constant, which extends the framework in Désir et al. (2020).

To overcome nonlinearity in the revenue maximization problem, the MILP formulation that we develop utilizes piecewise linear functions to approximate nonlinear functions. Various methods for constructing piecewise linear approximations of nonlinear functions have been proposed, see, for example, Markowitz and Manne (1957), Dantzig (1960), Croxton et al. (2003), Padberg (2000). Two well-known mixed-integer formulations for piecewise linear functions are the incremental cost (Markowitz and Manne 1957) and the convex combination formulations (Dantzig 1960). Alternatively, Beale and Tomlin (1970) suggest a formulation utilizing a class of constraints called special ordered sets of type 2 (SOS2). We refer the reader to Vielma et al. (2010) for a recent review of this topic. We use these results to develop a MILP formulation of the assortment optimization problem, and bound the error of the resulting approximation.

We develop an upper bound based on the well known Lagrangian relaxation approach. Lagrangian relaxation originated in the 1970s, see Held and Karp (1970), Fisher (1973) and Fisher (1981) for some early references, and has since been successfully used in numerous applications. In particular, Lagrangian relaxation has been successfully applied to revenue management, see Topaloglu (2009). Our Lagrangian upper bound is closely related to the one in Feldman and Topaloglu (2015a), where the authors bound the optimal expected revenue of the mixture-of-logits model as well as the mixture-of-nested-logit model. A key distinction of our work is that the Lagrangian upper bound in this work can be used when the number of candidate items is large, whereas Feldman and Topaloglu (2015a) assume a small number of candidate items; in addition, our work also incorporates display capacity constraints.

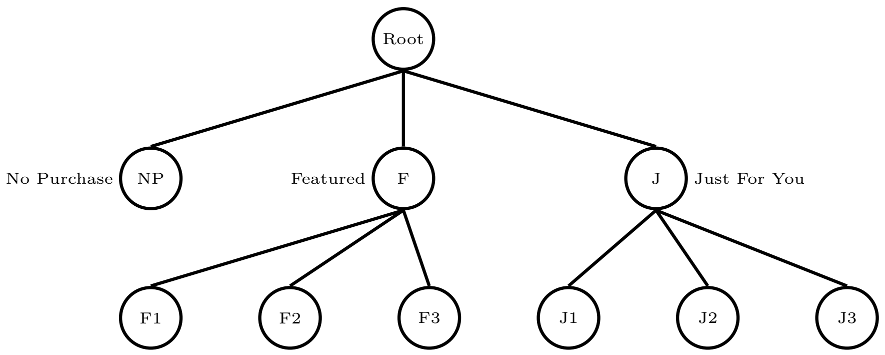
### 4.1.3 Organization

The remainder of the chapter is organized as follows. In Section 4.2, we describe our model setup and introduce certain preliminary results. In Section 4.3, we propose a FPTAS for a fixed number of customer segments. In Section 4.4, we propose a mixed integer linear programming formulation and show its theoretical performance. In Section 4.5, we introduce a fast heuristic solution algorithm and show its approximation guarantee. In Section 4.6, we develop a Lagrangian upper bound. In Section 4.7 we conduct an extensive numerical study. Section C.2 contains all technical proofs while Section C.1 illustrates other examples with the Featured vs JFY structure. In Section 4.8 we provide concluding remarks and potential future research.

## 4.2. Problem Formulation and Preliminaries

**Problem Statement.** Consider the store of an online multiplayer video game with a Featured section and a JFY section. Let $\mathcal{N}^F = \{1, ..., |\mathcal{N}^F|\}$ be the set of all candidate items of the Featured

Figure 4.3: Illustration of the Nested Logit Choice Model



section, and $\mathcal{N}^J = \{1, ..., |\mathcal{N}^J|\}$ be the set of all candidate JFY items. For item $i \in \mathcal{N}^F$, let $r_i^F \in \mathbb{R}_+$ be its unit price; for item $j \in \mathcal{N}^J$, let $r_j^J \in \mathbb{R}_+$ be its unit price. Both Featured and JFY sections have a display capacity $K \in \mathbb{Z}_+$; this requirement is naturally motivated by the layout of store interfaces. Let $\mathcal{S} = \{1, ..., |\mathcal{S}|\}$ be the set of customer segments, for $s \in \mathcal{S}$, let $m_s \in \mathbb{R}_+$ be the arrival rate of segment $s$ customers such that $\sum_{s \in \mathcal{S}} m_s = 1$. We assume that customers in the same segment are given the same JFY assortments. Throughout this chapter we reserve $i$ and $j$ and $s$ for $\mathcal{N}^F$, $\mathcal{N}^J$ and $\mathcal{S}$ and omit writing the index sets for better exposition.

For all $i \in \mathcal{N}^F$, define $a_i^F = 1$ if item $i$ is included in the Featured section and $a_i^F = 0$ otherwise; similarly for all $j \in \mathcal{N}^J$, define $a_{sj}^J = 1$ if item $j$ is included in the JFY section for customer segment $s$ and 0 otherwise. Let $\mathbf{a}^F = (a_1^F, ..., a_{|\mathcal{N}^F|}^F)$ be the Featured assortment vector, let $\mathbf{a}_s^J = (a_{s1}^J, ..., a_{s|\mathcal{N}^J|}^J)$ be the JFY assortment vector for segment $s$, and $\mathbf{a}^J = (\mathbf{a}_1^J, ..., \mathbf{a}_{|\mathcal{S}|}^J)$ represents the JFY assortments for all segments. Note that we use vector $\mathbf{a}_s^J$ to represent the JFY assortment for segment $s$, and $\mathbf{a}^J$ is a vector of vectors representing JFY assortments for all segments.

Under the nested logit model, an arriving customer decides to either make a purchase in one of the two nests (Featured and JFY), or leave without purchase; see Figure 4.3 for an illustration. If the customer decides to make a purchase in one of the nests, he or she has to choose one of the offered items, in other words the no purchase option is not available *within each nest*.

For customer segment $s \in \mathcal{S}$, let $v_{si}^F$ be the preference weight of item $i$ in the Featured section for customer segment $s$, and $v_{sj}^J$ is the preference weight of item $j$ in the JFY section for segment

$s$. Under the nested logit model, given that a segment $s$ customer decides to make a purchase in Featured, if the Featured assortment is $\mathbf{a}^F$, the probability that the customer chooses product $i'$ is given by $v^F_{si'} a^F_{i'} / \sum_i v^F_{si} a^F_i$. We normalize preference weights such that the preference of not making a purchase is 1 for all customer segments. For segment $s$ customers, The Featured nest has an associated dissimilarity parameter $\gamma^F_s$, characterizing the degree of dissimilarity of the products that can be offered in this nest; similarly let $\gamma^J_s$ be the dissimilarity parameter associated with the JFY nest for segment $s$ customers. If we offer $(\mathbf{a}^F, \mathbf{a}^J)$, then the probability that a segment $s$ customer decides to make a purchase in Featured is $(\sum_i v^F_{si} a^F_i)^{\gamma^F_s} / \left(1 + (\sum_i v^F_{si} a^F_i)^{\gamma^F_s} + (\sum_j v^J_{sj} a^J_j)^{\gamma^J_s}\right)$. This form of choice probabilities can be derived by using a random utility maximization principle; see McFadden et al. (1973) and Train (2003) for more details. In this work, we assume that $\gamma^F_s \in (0, 1]$ and $\gamma^J_s \in (0, 1]$ for all $s$.

It is worth noting that our assumptions that the no purchase option is not available within each nest and that dissimilarities are between zero and one may seem somewhat restrictive. In Davis et al. (2014), the authors study assortment optimization problems that allow both assumptions to be relaxed. We do not allow the no purchase option within nests mainly because it is empirically difficult to distinguish between the events of no purchase on the nest level and within nests; this assumption is common in the literature, see Gallego and Topaloglu (2014) and Chen et al. (2021b). Our assumption that dissimilarities are between zero and one is also quite commonly used in assortment planning and is a sufficient condition to guarantee compatibility with utility maximizing behavior, see McFadden et al. (1978), Gallego and Topaloglu (2014).

Under the mixture-of-nested-logit choice model, when the Featured assortment vector is $\mathbf{a}^F$ and the JFY assortment vector is $\mathbf{a}^J$, the total expected revenue from the store can be written as

$$\Pi(\mathbf{a}^F, \mathbf{a}^J) := \sum_s m_s \frac{\left(\sum_i a^F_i v^F_{si}\right)^{\gamma^F_s - 1} \sum_i a^F_i r^F_i v^F_{si} + \left(\sum_j a^J_{sj} v^J_{sj}\right)^{\gamma^J_s - 1} \sum_j a^J_{sj} r^J_j v^J_{sj}}{1 + \left(\sum_i a^F_i v^F_{si}\right)^{\gamma^F_s} + \left(\sum_j a^J_{sj} v^J_{sj}\right)^{\gamma^J_s}}.$$

Define the set of feasible assortment by

$$\mathcal{A} = \{(\mathbf{a}^F, \mathbf{a}^J) : a^F_i \in \{0,1\}, \forall i, a^J_{sj} \in \{0,1\}, \forall s, j, 1 \le \sum_i a^F \le K, 1 \le \sum_j a^J_s \le K, \forall s\}.$$

Note that in the above we require each section to include at least one item to avoid numerical error of 0 raised to a negative power. The assortment optimization problem can be written as

$$Z^* = \max_{(\mathbf{a}^F, \mathbf{a}^J) \in \mathcal{A}} \Pi(\mathbf{a}^F, \mathbf{a}^J), \tag{4.1}$$

where $Z^*$ is the optimal total expected revenue.

**Hardness Results.** Observe that the mixture-of-logits model reduces to problem (4.1) if we fix $\gamma_s^F = 1$ for all segments, $r_j^J = 0$ for all JFY items, and take $K = \max(|\mathcal{N}^F|, |\mathcal{N}^J|)$. As a result, solving (4.1) is at least as difficult as finding an assortment to offer that maximizes the expected revenue under mixture-of-logits, which is shown to be NP-Complete even when there are only two customer segments in Rusmevichientong et al. (2014). Recently, Désir et al. (2020) show that assortment optimization the mixture-of-logits model is hard to approximate within a reasonable factor. In the rest of this section, we introduce some relevant results from literature upon which we build our approaches.

### 4.2.1 Fix Featured Solve JFY

We begin introducing our approaches by making an important observation: one major complicating factor of problem (4.1) is the Featured assortment; indeed, if the Featured section is fixed, the problem becomes separable for each customer segment. In what follows, we introduce results from the literature that can be used to determine optimal segment-wise JFY assortments given a fixed Featured section.

**Theorem 4.1.** *(Fix Featured Solve JFY) Consider a variant of problem* (4.1)*, where the Featured section is fixed at $\hat{\mathbf{a}}^F$. Then an optimal JFY assortment $\hat{\mathbf{a}}^J = (\hat{\mathbf{a}}_1^J, ..., \hat{\mathbf{a}}_{|\mathcal{S}|}^J)$ can be obtained in polynomial time.*

*Proof.* See Section C.2. ■

Theorem 4.1 combines results from Rusmevichientong et al. (2010) and Gallego and Topaloglu (2014), and allows us to efficiently solve the revenue maximization problem for all segments when given a fixed Featured section. The detailed steps of obtaining optimal JFY assortments with a fixed Featured section are relegated to the Section C.2. We will refer to the procedure in Theorem 4.1 as `FixFeaturedSolveJFY`, which takes a Featured assortment as input and returns an optimal JFY assortment. The `FixFeaturedSolveJFY` procedure can be easily implemented when online multiplayer video game stores decide to take over the Feature assortment decision, and is also useful in developing our approaches later.

### 4.2.2 Reformulation and Structural Properties

The fact that problem (1) is easy to solve when the Featured section is fixed motivates us to view the problem as a two-stage optimization problem, of which the main challenge is finding a good Featured assortment. For ease of exposition, we introduce a reformulation of (4.1). For all $s \in \mathcal{S}$, define $\alpha_s(\mathbf{a}^F) = \sum_i a_i^F v_{si}^F$ and $\beta_s(\mathbf{a}^F) = \sum_i a_i^F v_{si}^F r_i^F$. We use $\alpha_s$ and $\beta_s$ without the dependency on $\mathbf{a}^F$ to represent our guesses on the values of $\alpha_s(\mathbf{a}^F)$ and $\beta_s(\mathbf{a}^F)$ without knowing $\mathbf{a}^F$. Define $z_s(\alpha_s, \beta_s)$ as

$$z_s(\alpha_s, \beta_s) = \max_{\mathbf{a}_s^J : 1 \le \sum_j a_{sj}^J \le K} \frac{\alpha_s^{\gamma_s^F - 1} \beta_s + \left(\sum_j a_{sj}^J v_{sj}^J\right)^{\gamma_s^J - 1} \sum_j a_{sj}^J r_j^J v_{sj}^J}{1 + \alpha_s^{\gamma_s^F} + \left(\sum_j a_{sj}^J v_{sj}^J\right)^{\gamma_s^J}}. \tag{4.2}$$

In other words, $z_s(\alpha_s, \beta_s)$ is the optimal total expected revenue from a segment $s$ customer when the Featured assortment $\mathbf{a}^F$ is chosen such that $\sum_i a_i^F v_{si}^F = \alpha_s$ and $\sum_i a_i^F v_{si}^F r_i^F = \beta_s$.

**Lemma 4.2.** *The assortment optimization problem* (4.1) *is equivalent to*

$$(\boldsymbol{OPT}) \quad Z^* = \max_{\mathbf{a}^F} \quad \sum_{s \in \mathcal{S}} m_s z_s(\alpha_s, \beta_s)$$

$$\alpha_s = \sum_i a_i^F v_{si}^F, \quad \forall s \in \mathcal{S},$$

$$\beta_s = \sum_i a_i^F r_i^F v_{si}^F, \quad \forall s \in \mathcal{S},$$

$$1 \le \sum_i a_i^F \le K,$$

$$a_i^F \in \{0, 1\}, \forall i \in \mathcal{N}^F,$$

*where $z_s(\alpha_s, \beta_s)$ is defined in* (4.2)*. Moreover, $z_s(\alpha_s, \beta_s)$ can be obtained in polynomial time.*

*Proof.* See Section C.2. ■

The reformulation is motivated by the fact that the optimal JFY assortments are easy to obtain when the Featured section is determined. With the reformulation, the problem can be viewed as finding the best $\alpha_s$ and $\beta_s$ values such that they correspond to a feasible Featured assortment, and at the same time maximize the total expected revenue. This idea drives both our FPTAS and the MILP approaches. Before we proceed to the details of the approaches, we first introduce some important properties of $z_s(\alpha_s, \beta_s)$ which will be useful for showing approximation results.

**Lemma 4.3.** *For all segments $s \in \mathcal{S}$, let $\alpha_{sL}$ and $\beta_{sL}$ be positive lower bounds on $\sum_i a_i^F v_{si}^F$ and $\sum_i a_i^F v_{si}^F r_i^F$, and let $\alpha_{sU}$ and $\beta_{sU}$ be upper bounds on $\sum_i a_i^F v_{si}^F$ and $\sum_i a_i^F v_{si}^F r_i^F$. Denote $\Delta_s = \{(\alpha_s, \beta_s) : \alpha_{sL} \leq \alpha_s \leq \alpha_{sU}, \beta_{sL} \leq \beta_s \leq \beta_{sU}\}$. Then for $z_s(\alpha_s, \beta_s)$ we have the following*

*(1) $z_s(\alpha_s, \beta_s)$ is continuous on $\Delta_s$,*

*(2) $z_s(\alpha_s, \beta_s)$ is decreasing in $\Delta_s$ in $\alpha_s$,*

*(3) $z_s(\alpha_s, \beta_s)$ is increasing in $\Delta_s$ in $\beta_s$,*

*(4) For any $c_1 \geq 1, 0 \leq c_2 \leq 1$ and $(\alpha_s, \beta_s)$ such that $(\alpha_s, \beta_s) \in \Delta_s$ and $(c_1\alpha_s, c_2\beta_s) \in \Delta_s$, we have $c_1/c_2 \cdot z_s(c_1\alpha_s, c_2\beta_s) \geq z_s(\alpha_s, \beta_s)$,*

*(5) For any $(\alpha_s, \beta_s)$ and $(\alpha'_s, \beta'_s)$ such that $\alpha_s \leq \alpha'_s \leq (1+\epsilon)\alpha_s$ and $\frac{1}{(1+\epsilon)}\beta_s \leq \beta'_s \leq \beta_s$, it holds that $z_s(\alpha'_s, \beta'_s) \leq z_s(\alpha_s, \beta_s) \leq (1+\epsilon)^2 z_s(\alpha'_s, \beta'_s)$.*

*Proof.* See Section C.2 ■

It is worth noting that obtaining upper and lower bounds on $\alpha_s$ and $\beta_s$ is straightforward. For instance, one could use $\alpha_{sL} = \min_i v_{si}^F, \alpha_{sU} = K \cdot \max_i v_{si}^F, \beta_{sL} = \min_i v_{si}^F r_i^F, \beta_{sU} = K \cdot \max_i v_{si}^F r_i^F$. This helps restrain the search on parameter values on a bounded region in which desired properties of $z_s(\cdot)$ hold. Intuitively, property 5 in Lemma 4.3 states that if two Featured assortments are such that their corresponding $\alpha_s$ and $\beta_s$ values are close for all segments, then their overall objective values are close. This property ensures that carefully chosen grid points on $\alpha_s$ and $\beta_s$ can lead to well approximated solutions. We expand on this idea and design an FPTAS in the next section.

## 4.3. Fully Polynomial-Time Approximation Scheme

The FPTAS is motivated by Lemma 4.3, in particular, the fact that the properties of $z_s(\alpha_s, \beta_s)$ ensure carefully chosen grid points on $\alpha_s$ and $\beta_s$ lead to well approximated solutions. In what follows, we highlight the main ideas of the FPTAS before describing the detailed steps. In particular, we make an connection with the multidimensional knapsack problem and use ideas from the knapsack literature to facilitate our approach. This type of frameworks have been adopted by Désir et al. (2020) and Liu et al. (2020) recently. Our approach relies on reformulation (**OPT**) and extends this line of work.

Note that assortment optimization of the mixture-of-logits model, which is a special case of our problem, is shown to be hard to approximate when the number of customer segments is not constant in Désir et al. (2020). As a result, our FPTAS assumes that the number of customer segments is constant, and achieves $(1-\epsilon)$ revenue with a running time polynomial in instance parameters and $1/\epsilon$.

**Parameter Grids.** Let $\boldsymbol{\alpha} = (\alpha_{sg}, s = 1, ..., |\mathcal{S}|, g = 1, ..., G_s)$ with $\alpha_{sL} < \alpha_{s1} \leq \alpha_{s2} \leq \cdots \leq \alpha_{sG} < \alpha_{sU}, \forall s \in \mathcal{S}$, and $\boldsymbol{\beta} = (\beta_{si}, s = 1, ..., |\mathcal{S}|, h = 1, ..., H_s)$ with $\beta_{sL} < \beta_{s1} \leq \beta_{s2} \leq \cdots \leq \beta_{sI} < \beta_{sU}, \forall s \in \mathcal{S}$ be grid points constructed in a way such that the ratio between adjacent points does not exceed $1+\epsilon$. Note that since $|\mathcal{S}|$ is fixed, there is a polynomial number of parameter grids. By creating the parameter grids, if we can find a Featured assortment for each parameter combination (or show that none exists) in polynomial time, the solution found can achieve near optimal revenue.

**Multidimensional Knapsack Subproblem.** For any parameter combination, which we denote $(\check{\alpha}_1, ..., \check{\alpha}_{|\mathcal{S}|}, \check{\beta}_1, ..., \check{\beta}_{|\mathcal{S}|})$, due to the monotonicity of $z_s(\alpha_s, \beta_s)$, if we could find a Featured assortment

$\mathbf{a}^F$ such that

$$\sum_i v_{si}^F a_i^F \leq \check{\alpha}_s, \quad \forall s, \tag{4.3}$$

$$\sum_i v_{si}^F a_i^F r_i^F \geq \check{\beta}_s, \quad \forall s, \tag{4.4}$$

$$1 \leq \sum_i a_i^F \leq K, \tag{4.5}$$

in polynomial time, or show that none exists, we have a FPTAS. Note that this subproblem can be viewed as finding a feasible solution to a $|\mathcal{S}|+1$ dimensional knapsack problem defined by (4.3) and (4.5), such that $|\mathcal{S}|$ distinct objective values defined by (4.4) all exceeds a certain threshold. One could use dynamic programming (DP) for multidimensional knapsack problems (see, e.g., Kellerer et al. 2004) for this task. It should be mentioned that the subproblems are completely independent and decomposable, making a distributed implementation straightforward.

**Dynamic Programming Subroutine.** It is well known that the DP approach for multidimensional knapsack problems is intractable, which means applying DP directly is prohibitively expensive computationally. To overcome this difficulty, we allow a small violation of the constraints, which in turn enables a discretization of the item "weights", yielding polynomial-sized state spaces for the DPs, and at the same time preserving approximation of solutions. To specify this idea, we introduce the following discretizations

$$w_{si} := \left\lceil \frac{K v_{si}^F}{\epsilon \check{\alpha}_s} \right\rceil, \quad w'_{si} := \left\lfloor \frac{K v_{si}^F r_i^F}{\epsilon \check{\beta}_s} \right\rfloor,$$

and replace constraint (4.3) and (4.4) with

$$\sum_i w_{si} a_i^F \leq \left\lceil \frac{K}{\epsilon} \right\rceil + K, \tag{4.6}$$

$$\sum_i w'_{si} a_i^F \geq \left\lfloor \frac{K}{\epsilon} \right\rfloor - K. \tag{4.7}$$

As we will show in the next Lemma, the discretizations defined above can be used to produce "approximately feasible" assortments.

**Lemma 4.4.** *(a) Any Featured assortment satisfying* (4.3) *and* (4.4) *will always satisfy* (4.6) *and* (4.7), *(b) any Featured assortment satisfying* (4.6) *will violate* (4.3) *by no more than a factor of* $(1+2\epsilon)$, *(c) any Featured assortment satisfying* (4.7) *will violate* (4.4) *by no more than a factor of* $1/(1-2\epsilon)$.

*Proof.* See Section C.2. ■

Note that after introducing the discretization, we are able to use DP to solve the resulting problems with a polynomial sized state space. Combined with (a)-(c), we are able to find a feasible Featured assortment that satisfies (4.3) and (4.4) approximately if one exists. To conclude the approximation scheme, the properties of $z_s(\alpha_s, \beta_s)$ can be used to show that a solution whose objective value is at least $\frac{1-2\epsilon}{(1+\epsilon)^2(1+2\epsilon)}$ times that of the true optimal solution is guaranteed.

**FPTAS.** We are now in position to describe detailed steps of the FPTAS.

**Algorithm 1** FPTAS for Online Multiplayer Gaming Assortment Optimization

```
Require: Instance data, $\epsilon > 0$
Ensure: Featured assortment $\hat{\mathbf{a}}^F$, JFY assortments $\hat{\mathbf{a}}^J$
  procedure Grid Points Construction
    for $s \in \mathcal{S}$ do
      $\alpha_{sL} \leftarrow \min_i v^F_{si}, \quad \alpha_{sU} \leftarrow K \cdot \max_i v^F_{si}, \quad \beta_{sL} \leftarrow \min_i v^F_{si} r^F_i, \quad \beta_{sU} \leftarrow K \cdot \max_i v^F_{si} r^F_i,$
      $G_s \leftarrow \lceil \log_{1+\epsilon}(\alpha_{sU}/\alpha_{sL}) \rceil + 1, \quad H_s \leftarrow \lceil \log_{1+\epsilon}(\beta_{sU}/\beta_{sL}) \rceil + 1,$
      $\alpha_{sg} \leftarrow \alpha_{sL} \cdot (1+\epsilon)^{g-1}, \forall g = 1, ..., G_s, \quad \beta_{sh} \leftarrow \beta_{sL} \cdot (1+\epsilon)^{h-1}, \forall h = 1, ..., H_s$
    end for
    $\mathscr{A} \leftarrow \bigtimes_s \{\alpha_{s1}, ..., \alpha_{sG_s}\}, \quad \mathscr{B} \leftarrow \bigtimes_s \{\beta_{s1}, ..., \beta_{sH_s}\}$
  end procedure
  procedure Dynamic Programming Subroutine
    $C \leftarrow \lceil K/\epsilon \rceil + K, \quad D \leftarrow \lfloor K/\epsilon \rfloor - K, \quad \mathscr{F} \leftarrow \emptyset$
    for $(\check{\alpha}_1, ..., \check{\alpha}_{|\mathcal{S}|}) \in \mathscr{A}, (\check{\beta}_1, ..., \check{\beta}_{|\mathcal{S}|}) \in \mathscr{B}$ do
      for $s \in \mathcal{S}, i \in \mathcal{N}^F$ do
        $w_{si} \leftarrow \lceil K v^F_{si} / (\epsilon \check{\alpha}_s) \rceil, \quad w'_{si} \leftarrow \lfloor K v^F_{si} r^F_i / (\epsilon \check{\beta}_s) \rfloor$
      end for
      for $i \in \mathcal{N}^F, c_1 \in \{0, ..., C\}, \cdots, c_{|\mathcal{S}|} \in \{0, ..., C\}, d_1 \in \{0, ..., D\}, \cdots, d_{|\mathcal{S}|} \in \{0, ..., D\}$ do
        $y(1, c_1, ..., c_{|\mathcal{S}|}, d_1, ..., d_{|\mathcal{S}|}) \leftarrow \begin{cases} 1, & \text{if } w_{s1} \le c_s \text{ and } w'_{s1} \ge d_s, \forall s, \\ 0, & \text{if } c_s \ge 0 \text{ and } d_s \le 0, \forall s, \\ \infty, & \text{otherwise,} \end{cases}$
        $y(i+1, c_1, ..., c_{|\mathcal{S}|}, d_1, ..., d_{|\mathcal{S}|}) \leftarrow$
          $\min\{y(i, c_1, ..., c_{|\mathcal{S}|}, d_1, ..., d_{|\mathcal{S}|}), 1 + y(i, c_1 - w_{1i}, ..., c_{|\mathcal{S}|} - w_{|\mathcal{S}|i}, d_1 - w'_{1i}, ..., d_{|\mathcal{S}|} - w'_{|\mathcal{S}|i})\}$
      end for
      if $y(|\mathcal{N}^F|, C, ..., C, D, ..., D) \le K$ then
        Let $\mathbf{a}^F$ be the Featured assortment corresponding to $y(|\mathcal{N}^F|, C, ..., C, D, ..., D)$, add $\mathbf{a}^F$ to $\mathscr{F}$
      end if
    end for
  end procedure
  procedure Finding Best Assortment
    $\hat{\mathbf{a}}^F \leftarrow \arg\max_{\mathbf{a}^F \in \mathscr{F}} \Pi \left(\mathbf{a}^F, \texttt{FixFeaturedSolveJFY}(\mathbf{a}^F)\right)$
    $\hat{\mathbf{a}}^J \leftarrow \texttt{FixFeaturedSolveJFY}(\hat{\mathbf{a}}^F)$
  end procedure
```

**Theorem 4.5.** *Let $(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J)$ be the solution found by Algorithm 1, it holds that*

$$\Pi(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J) \geq \frac{1-2\epsilon}{(1+\epsilon)^2(1+2\epsilon)} \cdot Z^*.$$

*Moreover, its running time is $\mathcal{O}\left((\log A_{\max})^{2|\mathcal{S}|} \cdot |\mathcal{N}^F| \cdot K^{2|\mathcal{S}|} \cdot (\frac{1}{\epsilon})^{4|\mathcal{S}|}\right)$, where*

$$A_{\max} := \max\{\max_s \frac{\alpha_{sU}}{\alpha_{sL}}, \max_s \frac{\beta_{sU}}{\beta_{sL}}\}.$$

*Proof.* See Section C.2. ∎

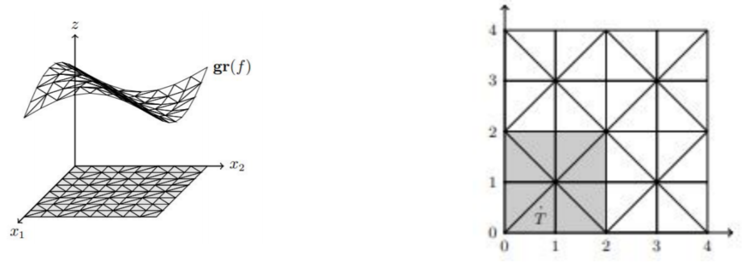
## 4.4. Piecewise Linear Approximate MILP

In this section, we propose a mixed integer linear programming (MILP) formulation for the assortment optimization problem. This approach is motivated by the numerical inefficiency of the FPTAS when the problem size increases. In order to achieve better performances, we create fewer grid points and represent function values by convex combinations of break point function values, in the hope of capturing the structural properties of $z_s(\cdot)$ by doing so.

Our MILP is also based on the reformulation presented in Lemma 4.2. We view (**OPT**) as a two-stage optimization problem, with the Featured assortment being the first stage decisions, and JFY assortments for all segments are the second stage decisions. Due to the nonlinearity of $z_s(\alpha_s, \beta_s)$, we proceed to approximate its value by creating breakpoints on $\alpha_s$ and $\beta_s$, and represent function values between breakpoints by convex combinations of function values of breakpoints. In particular, we adopt the "union jack trianglation" approach (Todd 1977), and approximate $z_s(\alpha_s, \beta_s)$ with a piecewise linear function that is linear in each triangle; see Figure 4.4 for a visual illustration.

As shown in Figure 4.4, for each triangle, we first calculate $z_s(\cdot)$ of its breakpoints, and for any point on this triangle, we represent its value by a convex combination of $z_s(\cdot)$ of the breakpoints. Then we create binary variables and impose `SOS2` constraints on convex combination weights to model this approximation; see Vielma and Nemhauser (2011) for details. We are now ready to state the MILP formulation.

Figure 4.4: Union Jack Triangulation (Todd 1977)



**Theorem 4.6.** *Let* $\boldsymbol{\alpha} = (\alpha_{sg}, s = 1, ..., |\mathcal{S}|, g = 1, ..., G_s)$ *with* $\alpha_{sL} < \alpha_{s1} \leq \alpha_{s2} \leq \cdots \leq \alpha_{sG} < \alpha_{sU}, \forall s \in \mathcal{S}$, *and* $\boldsymbol{\beta} = (\beta_{si}, s = 1, ..., |\mathcal{S}|, h = 1, ..., H_s)$ *with* $\beta_{sL} < \beta_{s1} \leq \beta_{s2} \leq \cdots \leq \beta_{sI} < \beta_{sU}, \forall s \in \mathcal{S}$ *be grid points constructed in a way such that the ratio between adjacent points does not exceed*

$1+\epsilon$. *Let* $\hat{\mathbf{a}}^F$ *be the optimal solution of the following MILP*

$$(\textit{PLA}) \quad \max \sum_{s\in\mathcal{S}} m_s y_s$$

$$s.t. \quad 1 \le \sum_i a_i^F \le K, \tag{4.8}$$

$$\sum_{g=1}^{G_s}\sum_{h=1}^{H_s} \mu_{sgh} = 1, \quad \forall s \in \mathcal{S}, \tag{4.9}$$

$$\nu_{sg} = \sum_{h=1}^{H_s} \mu_{sgh}, \quad \forall s \in \mathcal{S}, g = 1, ..., G_s, \tag{4.10}$$

$$\xi_{sh} = \sum_{g=1}^{G_s} \mu_{sgh}, \quad \forall s \in \mathcal{S}, h = 1, ..., H_s, \tag{4.11}$$

$$\{\nu_{s1}, ..., \nu_{sG}\} \ \texttt{SOS2}, \quad \forall s \in \mathcal{S}, \tag{4.12}$$

$$\{\xi_{s1}, ..., \xi_{sH}\} \ \texttt{SOS2}, \quad \forall s \in \mathcal{S}, \tag{4.13}$$

$$\sum_{g:\ odd} \sum_{h:\ even} \mu_{sgh} \le q_s, \quad \forall s \in \mathcal{S}, \tag{4.14}$$

$$\sum_{g:\ even} \sum_{h:\ odd} \mu_{sgh} \le 1 - q_s, \quad \forall s \in \mathcal{S}, \tag{4.15}$$

$$z_{sgh} = z_s(\alpha_{sg}, \beta_{sh}), \quad \forall s \in \mathcal{S}, g = 1, ..., G_s, h = 1, ..., H_s, \tag{4.16}$$

$$y_s = \sum_{g=1}^{G_s}\sum_{h=1}^{H_s} \mu_{sgh} z_{sgh}, \tag{4.17}$$

$$\sum_{g=1}^{G_s} \alpha_{sg}\nu_{sg} = \sum_i v_{si}^F a_i^F, \quad \forall s \in \mathcal{S}, \tag{4.18}$$

$$\sum_{h=1}^{H_s} \beta_{sh}\xi_{sh} = \sum_i v_{si}^F r_i^F a_i^F, \quad \forall s \in \mathcal{S}, \tag{4.19}$$

$$\mu_{sgi} \ge 0,\ q_s \ \textit{binary},\ a_i^F \ \textit{binary}, \quad \forall s, g, h, n. \tag{4.20}$$

*Let* $\hat{\mathbf{a}}^J$ *be the solution returned by* `FixFeaturedSolveJFY`$(\hat{\mathbf{a}}^F)$*, then we have*

$$\Pi(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J) \ge \frac{1}{(1+\epsilon)^4} Z^*.$$

*Proof.* See Section C.2. ∎

On a high level, (**PLA**) can be viewed as a guided search for a good Featured assortment. In the MILP, (4.8) is the display capacity constraint, (4.9)-(4.15) ensures the piecewise linear approximation coefficients are properly constructed, (4.16) calculates values at the breakpoints, (4.17) calculates the overall approximation, and (4.18)-(4.20) ensures feasibility of the featured assortment. By controlling the granularity of the parameter grid, (**PLA**) is able to achieve solutions with arbitrary level of quality.

It should be noted that the approximation results of the FPTAS and the MILP are based on the same properties of the $z_s(\cdot)$ function. One major difference of the two approaches is that in the FPTAS, we run a polynomially-sized DP for each parameter break point combination, and make no attempts to exploit the structural properties between break points, whereas in the MILP, we approximate the function value at any point by convex combinations its adjacent break points, allowing us to better represent the function values using fewer break points, even though by doing so we introduce binary decision variables which makes the approach non-polynomial. In our later numerical study, it is shown that the MILP outperforms the FPTAS empirically by a large margin. We believe this idea is a nice complement to the existing FPTAS and PTAS literature in revenue management, especially when the function to be optimized has properties similar to those in Lemma 4.3.

## 4.5. Heuristic Algorithm

The MILP approach and the FPTAS approach that we develop can be used to obtain solutions with arbitrary performance guarantees. However both approaches struggle when the number of customer segments $|\mathcal{S}|$ becomes large. In this section, we propose a fast heuristic that efficiently finds high quality solutions even when $|\mathcal{S}|$ is large. In addition, we provide an approximation bound for the heuristic.

Our heuristic follows the same idea in the FPTAS and the MILP: trying to find a good Featured assortment $\mathbf{a}^F$. To achieve this result, we consider each customer segment separately, and apply `StaticMNL` to produce candidate Featured assortments. According to Lemma C.2, these

Featured assortments have the potential to be optimal for an independent segment. The heuristic essentially “hopes” that one of these candidate Featured assortments is close to the optimal solution. We detail the procedure of the heuristic in Algorithm 2.

**Algorithm 2** Heuristic for Online Multiplayer Gaming Assortment Optimization

```
Require: Instance data
Ensure: Featured assortment $\hat{\mathbf{a}}^F$, JFY assortments $\hat{\mathbf{a}}^J$
  procedure INITIALIZATION
      Candidate Featured assortments $\mathscr{F} \leftarrow \emptyset$
  end procedure
  procedure CANDIDATE FEATURED ASSORTMENTS FOR INDEPENDENT SEGMENTS
      for $s \in \mathcal{S}$ do
          $\mathscr{F} \leftarrow \mathscr{F} \cup \texttt{StaticMNL}(\mathbf{v}_s^F, \mathbf{r}^F)$
      end for
  end procedure
  procedure BEST FEATURED ASSORTMENT IN CANDIDATES
      $\hat{\mathbf{a}}^F \leftarrow \arg\max_{\mathbf{a}^F \in \mathscr{F}} \Pi\left(\mathbf{a}^F, \texttt{FixFeaturedSolveJFY}(\mathbf{a}^F)\right)$
      $\hat{\mathbf{a}}^J \leftarrow \texttt{FixFeaturedSolveJFY}(\hat{\mathbf{a}}^F)$
  end procedure
```

We now introduce a theoretical result for the heuristic algorithm.

**Proposition 4.7.** *Let $\hat{\mathbf{a}}^F$ and $\hat{\mathbf{a}}^J$ be the solution returned by Algorithm 2. It holds that*

$$\Pi(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J) \geq \frac{1}{|\mathcal{S}|} \cdot Z^*$$

.

*Proof.* See Section C.2. ■

Note that the running time of the heuristic grows polynomially with the number of customer segments, which makes it efficient for large instances.

## 4.6. Upper Bounding the Optimal Expected Revenue

In this section, we propose a Lagrangian relaxation based upper bound. An upper bound is particularly useful when good quality solutions are easy to obtain, or when the problem is too hard to solve, and the only hope is heuristics. Our approach extends the results in Feldman and Topaloglu (2015a), where the authors develop an upper bound when there is only a small number of candidate items, while our bound can deal with hundreds of candidate items. The next lemma introduces a Lagrangian relaxation.

**Lemma 4.8.** *Let $\boldsymbol{\lambda} = (\lambda_{si} : s \in \mathcal{S}, i \in \mathcal{N}^F)$, and denote $\boldsymbol{\lambda}_s = (\lambda_{si} : i \in \mathcal{N}^F)$. Define $\pi_s(\boldsymbol{\lambda}_s)$ as the optimal objective value of the following linear program*

$$(\textit{LagSub}) \quad \max \; \frac{(\sum_i v^F_{si} a^F_i)^{\gamma^F_s - 1} \sum_i v^F_{si} a^F_i r^F_i + (\sum_j v^J_{sj} a^J_j)^{\gamma^J_s - 1} \sum_j v^J_{sj} a^J_j r^J_j}{1 + (\sum_i v^F_{si} a^F_i)^{\gamma^F_s} + (\sum_j v^J_{sj} a^J_j)^{\gamma^J_s}} - \sum_i \lambda_{si} a^F_i$$

$$\textit{s.t.} \quad 1 \le \sum_i a^F_i \le K,$$

$$1 \le \sum_j a^J_{sj} \le K,$$

$$a^F_i, a^J_{sj} \; \textit{binary}, \; \forall i, j.$$

*Then for any $\boldsymbol{\lambda}$ such that $\sum_s m_s \lambda_{si} = 0, \forall i$, we have $\sum_s m_s \pi_s(\boldsymbol{\lambda}_s) \ge Z^*$.*

*Proof.* See Section C.2. ■

The above lemma enables us to separate the problem by segments, we will construct upper bounds on $\pi_s(\boldsymbol{\lambda}_s)$ by conducting a grid search, which in turn yields valid upper bounds on $Z^*$. It is worth mentioning that setting $\lambda_{si} = 0, \forall s, i$ clearly satisfies $\sum_s m_s \lambda_{si} = 0, \forall i$, in which case $\sum_s m_s \pi_s(\boldsymbol{\lambda}_s)$ corresponds to the total expected revenue assuming that we are allowed to offer different Featured assortments for different segments of customers. We proceed to reformulate (**LagSub**) which will allow us to perform the grid search.

**Lemma 4.9.** *Problem (**LagSub**) is equivalent to the following reformulation*

$$\textbf{\textit{(LagSub-Reform)}} \quad \max \quad \frac{\alpha_s^{\gamma_s^F - 1}}{1 + \alpha_s^{\gamma_s^F} + \delta_s^{\gamma_s^J}} \sum_i v_{si}^F r_i^F a_i^F + \frac{\delta_s^{\gamma_s^J - 1}}{1 + \alpha_s^{\gamma_s^F} + \delta_s^{\gamma_s^J}} \sum_j v_{sj}^J r_j^J a_{sj}^J - \sum_i \lambda_{si}^F a_i^F$$

$$\alpha_s \geq \sum_i v_{si}^F a_i^F,$$

$$\delta_s \geq \sum_j v_{sj}^J a_{sj}^J,$$

$$1 \leq \sum_i a_i^F \leq K,$$

$$1 \leq \sum_j a_{sj}^J \leq K,$$

$$a_i^F, a_{sj}^J \ \textit{binary}, \forall i, j.$$

*Proof.* See Section C.2. ■

**Theorem 4.10.** *Define $\eta_s(\boldsymbol{\lambda}_s, \underline{\alpha}_s, \overline{\alpha}_s, \underline{\delta}_s, \overline{\delta}_s)$ with $\alpha_{sL} \leq \underline{\alpha}_s \leq \overline{\alpha}_s \leq \alpha_{sU}$ and $\delta_{sL} \leq \underline{\delta}_s \leq \overline{\delta}_s \leq \delta_{sU}$ as the optimal objective value of (**SubKnap-F**), where $\alpha_{sL}, \alpha_{sU}, \delta_{sL}$ and $\delta_{sU}$ are appropriately chosen bounds.*

$$(\textbf{\textit{SubKnap-F}}) \quad \max \quad \frac{\underline{\alpha}_s^{\gamma_s^F - 1}}{1 + \underline{\alpha}_s^{\gamma_s^F} + \underline{\delta}_s^{\gamma_s^J}} \sum_i v_{si}^F r_i^F a_i^F - \sum_i \lambda_{si}^F a_i^F$$

$$s.t. \quad \sum_i v_{si}^F a_i^F \leq \overline{\alpha}_s,$$

$$1 \leq \sum_i a_i^F \leq K,$$

$$0 \leq a_i^F \leq 1, \quad \forall i.$$

*Also define $\theta_s(\underline{\alpha}_s, \overline{\alpha}_s, \underline{\delta}_s, \overline{\delta}_s)$ as the optimal objective value of (**SubKnap-J**).*

$$\begin{aligned}
(\textbf{\textit{SubKnap-J}}) \quad \max \quad & \frac{\underline{\delta}_s^{\gamma_s^j - 1}}{1 + \underline{\alpha}_s^{\gamma_s^F} + \underline{\delta}_s^{\gamma_s^J}} \sum_j v_{sj}^J r_j^J a_{sj}^J \\
s.t. \quad & \sum_j v_{sj}^j a_{sj}^J \leq \overline{\delta}_s, \\
& 1 \leq \sum_i a_{sj}^J \leq K, \\
& 0 \leq a_{sj}^J \leq 1, \quad \forall j.
\end{aligned}$$

*Let $\boldsymbol{\alpha}_s = (\alpha_{s1}, ..., \lambda_{sG})$ and $\boldsymbol{\delta}_s = (\delta_{s1}, ..., \delta_{sP})$ with $\alpha_{sL} = \alpha_{s1} \leq \cdots \leq \alpha_{sG} = \alpha_{sU}$ and $\delta_{sL} = \delta_{s1} \leq \cdots \leq \delta_{sP} = \delta_{sU}$. Let*

$$\phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) = \max_{g,p} \Big( \eta_s(\boldsymbol{\lambda}_s, \alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1}) + \theta_s(\alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1}) \Big)$$

*then we have $\sum_s m_s \phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) \geq Z^*$.*

*Proof.* See Section C.2. ■

Theorem 4.10 enables us to obtain an upper bound with any $\boldsymbol{\lambda}$ such that $\sum_s m_s \lambda_{si} = 0, \forall i$ by solving two continuous knapsack subproblems for each grid, and pick the grid with the highest value for each segment. Note that (**SubKnap-F**) and (**SubKnap-J**) for each grid can be solve independently and efficiently, as a result parallel or distributed implementations could speed up the procedure significantly. Next we provide a subgradient method to iteratively obtain a good $\boldsymbol{\lambda}$.

**Lemma 4.11.** *$\phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s)$ is convex in $\boldsymbol{\lambda}_s$. For any $\hat{\boldsymbol{\lambda}}$ such that $\sum_s m_s \hat{\lambda}_{si} = 0, \forall i$, let $\hat{\mathbf{a}}_s^F$ be an optimal solution of (**SubKnap-F**) for segment $s$ at the grid where maximum is attained. A subgradient $\mathbf{d}_s = (d_{si} : i \in \mathcal{N}^F)$ of $\phi_s(\hat{\boldsymbol{\lambda}}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s)$ is given by $d_{si} = -\hat{a}_{si}^F, \forall i$.*

*Proof.* See Section C.2. ■

Theorem 4.10 and Lemma 4.11 allow us to run a subgradient routine to search for a good Lagrangian upper bound. The procedure of calculating the Lagrangian upper bound is as follows. First, initialize $\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s$ and $\boldsymbol{\delta}_s$ for all segments, and a step size scheduler; then for each segment and each grid, solve (**SubKnap-F**) and (**SubKnap-J**); for each segment, pick the grid with the highest $\phi_s(\cdot)$ value, calculate an upper bound for the incumbent $\boldsymbol{\lambda}$, and construct a subgradient; update $\boldsymbol{\lambda}$ following the step size scheduler and repeat until termination criterion is met. It is worth noting that in order to ensure $\sum_s m_s \lambda_{si} = 0, \forall i$ at all times, after each subgradient update, we randomly pick a segment $s'$ and set $\lambda_{s'i} = -(\sum_{s \in \mathcal{S}/\{s'\}} m_s \lambda_{si})/m_{s'}$ for all $i$.

The Lagrangian upper bound allows us to evaluate the quality of solutions found by the heuristic algorithm for large instances. It should also be noted that for small and medium sized instances, even though the FPTAS and MILP can theoretically produce solutions with arbitrary performance guarantee of $(1-\epsilon)$, choosing a larger $\epsilon$ can often lead to high quality solutions, in which case $(1-\epsilon)$ is not a good indicator of performances; the Lagrangian bound can be very useful in these situations as well, resulting in optimality gaps much tighter than $(1-\epsilon)$.

## 4.7. Numerical Study

In this section, we conduct numerical experiments on synthetic problem instances to test the performance of our proposed approaches under various settings.

### 4.7.1 Instance Data

Consider an online multiplayer video game store with a Featured section and a JFY section. Let $|\mathcal{S}|$ be the number of segments of its customers, $|\mathcal{N}^F|$ is the number of candidate Featured items, $|\mathcal{N}^J|$ is the number of candidate JFY items, and $K$ is the display capacity of both the Featured and JFY sections. In the numerical study, we generate 3 sizes of instances, `small`, `medium` and `large`; see Table 4.1 for a detailed description. Given any instance size configuration, we randomly generate instances in the following way. For each candidate Featured item $i$, its unit price $r_i^F$ is sampled from a uniform distribution $U(2, 10)$; for each candidate JFY item $j$, its unit price $r_j^J$ is sampled from a uniform distribution $U(0.2, 1)$; note that here we assume that the Featured items are more valuable than the JFY items. The arrival rate $m_s$ of each customer segment is sampled from a uniform distribution $U(0.5, 1)$ and then normalized. The dissimilarity parameters $\gamma_s^F$ and $\gamma_s^J$ are sampled from uniform distribution $U(0.25, 1)$. The preference weights of JFY items are randomly sampled from exponential distribution $Exp(0.005)$. The preference weights of the Featured items are chosen such that it is non-trivial to determine the Featured assortment.

Table 4.1: Assortment Optimization Instance Size Description

| Size | $\|\mathcal{S}\|$ | $\|\mathcal{N}^F\|$ | $\|\mathcal{N}^J\|$ | $K$ |
|---|---|---|---|---|
| `small` | 2 | 10 | 20 | 3 |
| `medium` | 10 | 30 | 100 | 5 |
| `large` | 20 | 100 | 200 | 10 |

### 4.7.2 Algorithms

We briefly describe the algorithms tested in the numerical study, as well as some noteworthy implementation details.

**FPTAS (FPT).** The procedure of the FPTAS is described in Algorithm 1. Note that if we wish to choose an $\epsilon$ such that the performance guarantee by Theorem 4.5 is practically useful, even for `small` instances, the procedure becomes too expensive. In our experiments, we use $\epsilon = 0.15$, which allows the procedure to terminate quickly and at the same time produce high quality assortments. For `medium` and `large` instances, the number of break point combinations becomes astronomical, and the FPTAS in turn becomes intractable.

**MILP (MIP).** The motivation of the MILP approach is to approximate the objective function using fewer break points relative to the FPTAS. In the numerical experiments, for `small` instances, we formulate (**PLA**) using 50 uniformly distributed break points for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$; for `medium` instances, we formulate (**PLA**) using 20 breakpoints for $\boldsymbol{\alpha}$ and 10 breakpoints for $\boldsymbol{\beta}$; the MILP approach becomes intractable for `large` instances. We use `Gurobi 9.0.3` under the default setting to solve the integer programs. During the branch and bound procedure, we implement a callback subroutine to evaluate the true objective function value whenever a new feasible Featured assortment is found, and return the solution (which may not be the optimal solution of the (**PLA**)) with the highest objective value when the branch and bound procedure terminates.

**Heuristic (HEU).** The heuristic described in Algorithm 2 is efficient across all sizes, and easy to implement without any tuning required.

**Greedy (GRD).** As a benchmark, we implement an intuitive greedy algorithm which successively add the most promising item to the Featured assortment, and update the corresponding optimal JFY assortments when necessary. It should be noted that the objective function does not enjoy the submodularity property, and therefore a theoretical performance guarantee of the greedy algorithm is hard to obtain. The greedy algorithm is also efficient across all sizes and easy to implement.

**Lagrangian Upper Bound (LUB).** It is well known that the subgradient procedure for calculating the Lagrangian upper bound requires carefully chosen step sizes. In our experiments, the step size of iteration $n$ is of the form $1/n$; more sophisticated step size schedulers may lead to better performances. In addition to the step size scheduler, we also need to choose the number of break points for $\boldsymbol{\alpha}$ and $\boldsymbol{\delta}$ for the Lagrangian upper bound. In our implementation, the break points are exponentially distributed, that is, the ratio of adjacent points is a constant. For `small` instances, we choose 400 break points for $\boldsymbol{\alpha}$ and 100 break points for $\boldsymbol{\delta}$; for `medium` instances, we use 400 break points for $\boldsymbol{\alpha}$ and 30 break points for $\boldsymbol{\delta}$; for `large` instances, we use 300 break points for $\boldsymbol{\alpha}$ and 30 break points for $\boldsymbol{\delta}$. The Lagrangian upper bound scales well for all instances, since each customer segment is treated independently. It is also well suited for a distributed implementation.

**Naive Upper Bound (NUB).** As a benchmark, we calculate a naive upper bound on the total expected revenue by treating each customer segment independently, and allow the Featured assortment to be different for different segments. This upper bound can be viewed as a Lagrangian upper bound without penalty, and is efficient across all instances.

### 4.7.3 Numerical Results

All experiments are run on a MacBook Pro with an intel i9 and 16 GB of RAM. All code is written in `Python 3.7.4`, and optimization models are solved using `Gurobi 9.1`. The average running time of the algorithms are reported in Table 4.2. Note that the store is usually periodically updated either daily or weekly, and all algorithms except for the Lagrangian upper bound are able to terminate within a few minutes even for large instances; the Lagrangian upper bound is also able to achieve better performance than the naive upper bound within a day on a personal computer, and with a distributed implementation the procedure can realistically speed up significantly, making it practically useful.

Table 4.2: Average Running Time for Assortment Optimization

| Time (s) | `small` | `medium` | `large` |
|---|---|---|---|
| FPT | 14.5 | n/a | n/a |
| MIP | 3.67 | 193.2 | n/a |
| HEU | 0.02 | 3.70 | 118.4 |
| GRD | 0.05 | 3.34 | 134.5 |
| LUB | 3600 | 7200 | 72000 |
| NUB | 0.005 | 0.62 | 8.66 |

**Small Instances.** We generate 100 `small` instances. For each instance, we run all of FPT, MIP, HEU, GRD, LUB and NUB, and summarize our results in Table 4.3. The left half of the table reports optimality gaps; specifically, we compare FPT, MIP and HEU solutions against the LUB, and also the GRD solution against the NUB, and calculate the average gap, the 95th percentile gap and the max gap for each pair. On the right half of the table, we calculate the relative improvements of our proposed approaches against the baselines; in particular, we compare the revenue improvement of FPT, MIP and HEU against the GRD baseline, as well the upper bound improvement of LUB against NUB; we also include the average improvement, 95th percentile improvement and the max improvement. From Table 4.3, it is clear that all FPT, MIP and HEU outperform the GRD baseline, and LUB is tighter than the NUB. The average gap between the GRD baseline and the NUB baseline is 3.31% aross 100 instances, with the max gap as large as 15.6%, whereas FPT, MIP and HEU all manage to achieve average gaps under 2% with the max gap being only 6%. Moreover, the reduction in optimality gaps primarily comes from revenue improvement, with an average improve around 1%, and the max revenue improvement as large as over 16%, which given the magnitude of video game store microtrasactions, implies a potentially significant revenue boost. Among FPT, MIP and HEU, we see that the MIP is the most competitive, achieving the best revenue performance; HEU is also very good, outperforming FPT with only a fraction of its running time. Finally, LUB is consistently better than the NUB baseline, averaging nearly 1% bound improvement, with the max being over 6% percent.

**Medium Instances.** We generate 30 `medium` instances, and report the results in Table 4.4. Note that FPT is no longer tractable for `medium` instances due to the increased number of customer segments. Overall, we observe similar patterns to the results of the `small` instances. The MIP continue to be the most competitive, reducing the baseline gap from 6.58% to 3.05% on average. The revenue improvement of MIP and HEU compared to the GRD is even more significant than that of `small` instances, with the an over 2% average improvement; the max improvement is over 30%, which means GRD could perform very poorly on some instances. The LUB again outperforms NUB, achieving a 1.53% tighter bound on average.

**Large Instances.** We generate 10 `large` instances, and report the results in Table 4.5. Note that even the MIP struggles to terminate for `large` instances. From Table 4.5 we see that HEU achieves better performance compared to the GRD baseline, reducing the gap slightly, and achieves a 0.72% average revenue improvement. The largest revenue improvement is 18.6%, which suggests HEU has the potential to avoid significant revenue loss. On the other hand, LUB once again outperforms the NUB.

**Summary of Results.** All of the approaches developed in this work achieve better performance than the baseline as expected. To find high quality solutions, using MIP whenever possible would result in the best revenue performance, while the HEU can produce good assortments when the size becomes too large for MIP. The FPT is not as practical especially when the number of customer segments increases, even though it comes with a nice theoretical guarantee. The Lagrangian bound LUB is consistently better than NUB, which helps evaluate feasible assortments found by various algorithms.

Table 4.3: Numerical results for `small` assortment optimization instances

| Percentage Gap (%) | | | | Relative Improvement (%) | | | |
|---|---|---|---|---|---|---|---|
| | Average | 95th | Max | | Average | 95th | Max |
| FPT/LUB | 1.82 | 3.40 | 6.00 | FPT vs. GRD | 0.74 | 5.79 | 16.62 |
| MIP/LUB | 1.49 | 3.30 | 6.00 | MIP vs. GRD | 1.08 | 5.79 | 16.62 |
| HEU/LUB | 1.66 | 3.39 | 6.00 | HEU vs. GRD | 0.91 | 5.79 | 16.62 |
| GRD/NUB | 3.31 | 8.47 | 15.6 | LUB vs. NUB | 0.84 | 3.41 | 6.31 |

## 4.8. Conclusions and Future Research

In this work, we study an assortment optimization problem for multiplayer online video games under discrete choice models. The game stores include a Featured section and a JFY section for marketing and operations purposes, and customer heterogeneity is characterized by customer segments. We propose a FPTAS as well as a MILP formulation that returns high quality solutions with arbitrary theoretical performance guarantees. We also design a heuristic algorithm for large instances with an approximation result, and a Lagrangian upper bound for evaluating feasible solutions. Our numerical experiments illustrate that our approaches perform well across various settings. Methodologically, our work is the first to devise solution algorithms for the mixture-of-nested-logit model, one of the most difficult classes of problems in assortment optimization.

Our work opens up a number of directions for future research. It would be interesting to see if the MILP formulation can be improved by using a more advanced technique with fewer binary variables, or apply a Benders decomposition to iteratively solve the formulation. One could also try to characterize conditions under which submodularity holds, in which case greedy heuristics are known to work well. It is always possible to analyze customers' choices under alternative discrete choice models. One unique feature of online multiplayer video game stores is that the complete purchase history and inventory information of all customers is available to the seller, which may lead to opportunities to better understand the resulting assortment optimization problem. We leave this for future research.

Table 4.4: Numerical results for `medium` assortment optimization instances

| Percentage Gap (%) | | | | Relative Improvement (%) | | | |
|---|---|---|---|---|---|---|---|
| | Average | 95th | Max | | Average | 95th | Max |
| MIP/LUB | 3.05 | 5.25 | 6.02 | MIP vs. GRD | 2.53 | 14.21 | 32.74 |
| HEU/LUB | 3.45 | 5.70 | 6.38 | HEU vs. GRD | 2.11 | 14.13 | 32.74 |
| GRD/NUB | 6.58 | 15.79 | 28.93 | LUB vs. NUB | 1.53 | 3.05 | 3.45 |

Table 4.5: Numerical results for `large` assortment optimization instances

| Percentage Gap (%) | | | | Relative Improvement (%) | | | |
|---|---|---|---|---|---|---|---|
| | Average | 95th | Max | | Average | 95th | Max |
| HEU/LUB | 5.68 | 6.78 | 7.77 | HEU vs. GRD | 0.72 | 6.64 | 18.60 |
| GRD/NUB | 6.26 | 11.25 | 19.62 | LUB vs. NUB | 1.83 | 2.66 | 2.76 |

# Bibliography

Abraham, David J, Avrim Blum, Tuomas Sandholm. 2007. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. *Proceedings of the 8th ACM conference on Electronic commerce.* ACM, 295–304.

Adelman, Daniel. 2007. Dynamic bid prices in revenue management. *Operations Research* **55**(4) 647–661.

Agarwal, Sharad, Jacob R Lorch. 2009. Matchmaking for online games and other latency-sensitive p2p systems. *Proceedings of the ACM SIGCOMM 2009 conference on Data communication.* 315–326.

Aldrich, John H, Forrest D Nelson. 1984. *Linear probability, logit, and probit models.* 45, Sage.

Alptekinoğlu, Aydın, John H Semple. 2016. The exponomial choice model: A new alternative for assortment and price optimization. *Operations Research* **64**(1) 79–93.

Ansari, Asim, Carl F Mela. 2003. E-customization. *Journal of marketing research* **40**(2) 131–145.

Aouad, Ali, Vivek Farias, Retsef Levi, Danny Segev. 2018. The approximability of assortment optimization under ranking preferences. *Operations Research* **66**(6) 1661–1669.

Aouad, Ali, Omer Saritac. 2019. Dynamic stochastic matching under limited time. *Available at SSRN: https://ssrn.com/* .

Arora, Neeraj, Xavier Dreze, Anindya Ghose, James D Hess, Raghuram Iyengar, Bing Jing, Yogesh Joshi, V Kumar, Nicholas Lurie, Scott Neslin, et al. 2008. Putting one-to-one marketing to work: Personalization, customization, and choice. *Marketing Letters* **19**(3-4) 305.

Awasthi, Pranjal, Tuomas Sandholm. 2009. Online stochastic optimization in the large: Application to kidney exchange. *IJCAI*, vol. 9. 405–411.

Banerjee, Siddhartha, Yash Kanoria, Pengyu Qian. 2018. State dependent control of closed queueing networks with application to ride-hailing. *arXiv preprint arXiv:1803.04959* .

Banerjee, Siddhartha, Carlos Riquelme, Ramesh Johari. 2015. Pricing in ride-share platforms: A queueing-theoretic approach. *Available at SSRN 2568258* .

Beale, Evelyn Martin Lansdowne, John A Tomlin. 1970. Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables. *OR* **69**(447-454) 99.

Ben-Akiva, Moshe E, Steven R Lerman, Steven R Lerman, et al. 1985. *Discrete choice analysis: theory and application to travel demand*, vol. 9. MIT press.

Bennett, James, Stan Lanning, et al. 2007. The netflix prize. *Proceedings of KDD cup and workshop*, vol. 2007. Citeseer, 35.

Berbeglia, Gerardo, Agustín Garassino, Gustavo Vulcano. 2021. A comparative empirical study of discrete choice models in retail operations. *Management Science* .

Bernstein, Fernando, A Gürhan Kök, Lei Xie. 2015. Dynamic assortment customization with limited inventories. *Manufacturing & Service Operations Management* **17**(4) 538–553.

Bernstein, Fernando, Sajad Modaresi, Denis Sauré. 2019. A dynamic clustering approach to data-driven assortment personalization. *Management Science* **65**(5) 2095–2115.

Bertsekas, Dimitri P. 2008. Neuro-dynamic programming. *Encyclopedia of optimization.* Springer, 2555–2560.

Blanchet, Jose, Guillermo Gallego, Vineet Goyal. 2016. A markov chain approximation to choice modeling. *Operations Research* **64**(4) 886–905.

Bobadilla, Jesús, Fernando Ortega, Antonio Hernando, Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* **46** 109–132.

Bray, M, W Wang, PX-K Song, AB Leichtman, MA Rees, VB Ashby, R Eikstadt, A Goulding, JD Kalbfleisch. 2015. Planning for uncertainty and fallbacks can increase the number of transplants in a kidney-paired donation program. *American Journal of Transplantation* **15**(10) 2636–2645.

Butcher, Chris. 2008. E pluribus unum: Matchmaking in halo 3. *Game Developers Conference (GDC 2008)*.

Chen, Le, Alan Mislove, Christo Wilson. 2015. Peeking beneath the hood of uber. *Proceedings of the 2015 Internet Measurement Conference*. ACM, 495–508.

Chen, M Keith, Michael Sheldon. 2016. Dynamic pricing in a labor market: Surge pricing and flexible work on the uber platform. *Ec*. 455.

Chen, Xi, Zachary Owen, Clark Pixton, David Simchi-Levi. 2021a. A statistical learning approach to personalization in revenue management. *Management Science* .

Chen, Xi, Zachary Owen, Clark Pixton, David Simchi-Levi. 2022. A statistical learning approach to personalization in revenue management. *Management Science* **68**(3) 1923–1937.

Chen, Xi, Chao Shi, Yining Wang, Yuan Zhou. 2021b. Dynamic assortment planning under nested logit models. *Production and Operations Management* **30**(1) 85–102.

Chen, Xi, Yining Wang, Yuan Zhou. 2020. Dynamic assortment optimization with changing contextual information. *Journal of machine learning research* .

Cheung, Wang Chi, David Simchi-Levi. 2017. Thompson sampling for online personalized assortment optimization problems with multinomial logit choice models. *Available at SSRN 3075658* .

Chiu, Chui-Yu, Yi-Feng Chen, I-Ting Kuo, He Chun Ku. 2009. An intelligent market segmentation system using k-means and particle swarm optimization. *Expert systems with applications* **36**(3) 4558–4565.

Cook, William J, WH Cunningham, WR Pulleyblank, A Schrijver. 2009. Combinatorial optimization. *Oberwolfach Reports* **5**(4) 2875–2942.

Croissant, Yves, et al. 2012. Estimation of multinomial logit models in r: The mlogit packages. *R package version 0.2-2. URL: http://cran. r-project. org/web/packages/mlogit/vignettes/mlogit. pdf* .

Croxton, Keely L, Bernard Gendron, Thomas L Magnanti. 2003. A comparison of mixed-integer programming models for nonconvex piecewise linear cost minimization problems. *Management Science* **49**(9) 1268–1273.

Dantzig, George B. 1960. On the significance of solving linear programming problems with some integer variables. *Econometrica, Journal of the Econometric Society* 30–44.

Davis, James M, Guillermo Gallego, Huseyin Topaloglu. 2014. Assortment optimization under variants of the nested logit model. *Operations Research* **62**(2) 250–273.

De Farias, Daniela Pucci, Benjamin Van Roy. 2003. The linear programming approach to approximate dynamic programming. *Operations research* **51**(6) 850–865.

De Farias, Daniela Pucci, Benjamin Van Roy. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of operations research* **29**(3) 462–478.

Derman, Cyrus, Gerald J Lieberman, Sheldon M Ross. 1972. A sequential stochastic assignment problem. *Management Science* **18**(7) 349–355.

Désir, Antoine, Vineet Goyal, Srikanth Jagabathula, Danny Segev. 2021. Mallows-smoothed distribution over rankings approach for modeling choice. *Operations Research* .

Désir, Antoine, Vineet Goyal, Jiawei Zhang. 2020. Capacitated assortment optimization: Hardness and approximation. *Available at SSRN 2543309* .

Desrosiers, Jacques, Marco E Lübbecke. 2005. A primer in column generation. *Column generation*. Springer, 1–32.

Dickerson, John P, Ariel D Procaccia, Tuomas Sandholm. 2012. Dynamic matching via weighted myopia with application to kidney exchange. *AAAI*.

Dickerson, John P, Ariel D Procaccia, Tuomas Sandholm. 2018. Failure-aware kidney exchange. *Management Science* .

Edmonds, Jack. 1965. Paths, trees, and flowers. *Canadian Journal of mathematics* **17**(3) 449–467.

Elo, Arpad E. 1978. *The rating of chessplayers, past and present.* Arco Pub.

Emek, Yuval, Shay Kutten, Roger Wattenhofer. 2016. Online matching: haste makes waste! *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing.* ACM, 333–344.

Erdos, Paul, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci* **5**(1) 17–60.

Farias, Vivek F, Benjamin Van Roy. 2007. An approximate dynamic programming approach to network revenue management. *Preprint* .

Feldman, Jacob, Huseyin Topaloglu. 2015a. Bounding optimal expected revenues for assortment optimization under mixtures of multinomial logits. *Production and Operations Management* **24**(10) 1598–1620.

Feldman, Jacob, Dennis J Zhang, Xiaofei Liu, Nannan Zhang. 2022. Customer choice models vs. machine learning: Finding optimal product displays on alibaba. *Operations Research* **70**(1) 309–328.

Feldman, Jacob B, Huseyin Topaloglu. 2015b. Capacity constraints across nests in assortment optimization under the nested logit model. *Operations Research* **63**(4) 812–822.

Feldman, Jon, Aranyak Mehta, Vahab Mirrokni, Shan Muthukrishnan. 2009. Online stochastic matching: Beating 1-1/e. *2009 50th Annual IEEE Symposium on Foundations of Computer Science.* IEEE, 117–126.

Feng, Qi, J George Shanthikumar, Mengying Xue. 2022. Consumer choice models and estimation: A review and extension. *Production and Operations Management* **31**(2) 847–867.

Fisher, Marshall L. 1973. Optimal solution of scheduling problems using lagrange multipliers: Part i. *Operations Research* **21**(5) 1114–1127.

Fisher, Marshall L. 1981. The lagrangian relaxation method for solving integer programming problems. *Management science* **27**(1) 1–18.

Frederickson, Ben. 2022. Implicit: Fast python collaborative filtering for implicit datasets. URL `https://benfred.github.io/implicit/`.

Gallego, Guillermo, Huseyin Topaloglu. 2014. Constrained assortment optimization for the nested logit model. *Management Science* **60**(10) 2583–2601.

Gallego, Guillermo, Huseyin Topaloglu, et al. 2019. *Revenue management and pricing analytics*, vol. 209. Springer.

Glorie, Kristiaan M, J Joris van de Klundert, Albert PM Wagelmans. 2014. Kidney exchange with long chains: An efficient pricing algorithm for clearing barter exchanges with branch-and-price. *Manufacturing & Service Operations Management* **16**(4) 498–512.

Gomory, Ralph E, Tien Chung Hu. 1961. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics* **9**(4) 551–570.

Goyal, Vineet, Noemie Perivier. 2021. Dynamic pricing and assortment under a contextual mnl demand. *arXiv preprint arXiv:2110.10018* .

Greene, William H. 2003. *Econometric analysis.* Pearson Education India.

Greene, William H, David A Hensher. 2003. A latent class model for discrete choice analysis: contrasts with mixed logit. *Transportation Research Part B: Methodological* **37**(8) 681–698.

Hastie, Trevor, Robert Tibshirani, Jerome H Friedman, Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer.

Held, Michael, Richard M Karp. 1970. The traveling-salesman problem and minimum spanning trees. *Operations Research* **18**(6) 1138–1162.

Herbrich, Ralf, Tom Minka, Thore Graepel. 2007. Trueskill™: a bayesian skill rating system. *Advances in neural information processing systems*. 569–576.

Hu, Ming, Yun Zhou. 2022. Dynamic type matching. *Manufacturing & Service Operations Management* **24**(1) 125–142.

Hu, Yifan, Yehuda Koren, Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.

Jagabathula, Srikanth, Lakshminarayanan Subramanian, Ashwin Venkataraman. 2018. A model-based embedding technique for segmenting customers. *Operations Research* **66**(5) 1247–1267.

Kallus, Nathan, Madeleine Udell. 2020. Dynamic assortment personalization in high dimensions. *Operations Research* .

Kanoria, Yash, Pengyu Qian. 2019. Near optimal control of a ride-hailing platform via mirror backpressure. *arXiv preprint arXiv:1903.02764* .

Kansal, Tushar, Suraj Bahuguna, Vishal Singh, Tanupriya Choudhury. 2018. Customer segmentation using k-means clustering. *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*. IEEE, 135–139.

Karp, Richard M, Umesh V Vazirani, Vijay V Vazirani. 1990. An optimal algorithm for on-line bipartite matching. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. ACM, 352–358.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer, Berlin, Germany.

Keskin, N Bora, Yuexing Li, Nur Sunar. 2022. Data-driven clustering and feature-based retail electricity pricing with smart meters. *Available at SSRN 3686518* .

Kök, A Gürhan, Marshall L Fisher. 2007. Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research* **55**(6) 1001–1021.

Kök, A Gürhan, Marshall L Fisher, Ramnath Vaidyanathan. 2008. Assortment planning: Review of literature and industry practice. *Retail supply chain management* 99–153.

Kök, A Gürhan, Yi Xu. 2011. Optimal and competitive assortments with endogenous pricing under hierarchical consumer choice models. *Management Science* **57**(9) 1546–1563.

Kuo, RJ, LM Ho, Clark M Hu. 2002. Integration of self-organizing feature map and k-means algorithm for market segmentation. *Computers & Operations Research* **29**(11) 1475–1493.

Lancaster, Kelvin J. 1966. A new approach to consumer theory. *Journal of political economy* **74**(2) 132–157.

Lee, Joonkyum, Vishal Gaur, Suresh Muthulingam, Gary F Swisher. 2016. Stockout-based substitution and inventory planning in textbook retailing. *Manufacturing & Service Operations Management* **18**(1) 104–121.

Lei, Yanzhe, Stefanus Jasin, Joline Uichanco, Andrew Vakhutinsky. 2021. Joint product framing (display, ranking, pricing) and order fulfillment under the multinomial logit model for e-commerce retailers. *Manufacturing & Service Operations Management* .

Li, Zhuoshu, Kelsey Lieberman, William Macke, Sofia Carrillo, Chien-Ju Ho, Jason Wellen, Sanmay Das. 2019. Incorporating compatible pairs in kidney exchange: A dynamic weighted matching model. *EC*. 349–367. URL `https://doi.org/10.1145/3328526.3329619`.

Lin, Qihan, Selvaprabu Nadarajah, Negar Soheili. 2020. Revisiting approximate linear programming: Constraint-violation learning with applications to inventory control and energy storage. *Management Science* **66**(4) 1509–1782.

Linden, Greg, Brent Smith, Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* **7**(1) 76–80.

Liu, Nan, Yuhang Ma, Huseyin Topaloglu. 2020. Assortment optimization under the multinomial logit model with sequential offerings. *INFORMS Journal on Computing* **32**(3) 835–853.

Luce, R. 1959. *Individual Choice Behavior*. John Wiley & Sons, New York.

Lyu, Chengyi, Stefanus Jasin, Sajjad Najafi, Huanan Zhang. 2021. Assortment optimization with multi-item basket purchase under multivariate mnl model. *Available at SSRN 3818886* .

Manweiler, Justin, Sharad Agarwal, Ming Zhang, Romit Roy Choudhury, Paramvir Bahl. 2011. Switchboard: a matchmaking system for multiplayer mobile games. *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 71–84.

Markowitz, Harry M, Alan S Manne. 1957. On the solution of discrete programming problems. *Econometrica: journal of the Econometric Society* 84–110.

McFadden, Daniel. 1981. Econometric models of probabilistic choice. *Structural analysis of discrete data with econometric applications* **198272**.

McFadden, Daniel, Kenneth Train. 2000. Mixed mnl models for discrete response. *Journal of applied Econometrics* **15**(5) 447–470.

McFadden, Daniel, et al. 1973. Conditional logit analysis of qualitative choice behavior .

McFadden, Daniel, et al. 1978. Modelling the choice of residential location .

Mehta, Aranyak, Amin Saberi, Umesh Vazirani, Vijay Vazirani. 2007. Adwords and generalized online matching. *Journal of the ACM (JACM)* **54**(5) 22.

Meissner, Joern, Arne Strauss. 2012. Network revenue management with inventory-sensitive bid prices and customer choice. *European Journal of Operational Research* **216**(2) 459–468.

Miao, Sentao, Xiuli Chao. 2019. Fast algorithms for online personalized assortment optimization in a big data regime. *Available at SSRN 3432574* .

Miao, Sentao, Xi Chen, Xiuli Chao, Jiaxi Liu, Yidong Zhang. 2019. Context-based dynamic pricing with online clustering. *arXiv preprint arXiv:1902.06199* .

Newzoo. 2020. Three billion players by 2023: Engagement and revenues continue to thrive across the global games market. URL `https://newzoo.com/insights/articles/games-market-engagement-revenues-trends-2020-2023-gaming-report`.

Özkan, Erhun, Amy R Ward. 2020. Dynamic matching for real-time ride sharing. *Stochastic Systems* **10**(1) 29–70.

Padberg, Manfred. 2000. Approximating separable nonlinear functions via mixed zero-one programs. *Operations Research Letters* **27**(1) 1–5.

Padberg, Manfred W, M Ram Rao. 1982. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research* **7**(1) 67–80.

Patrick, Jonathan, Martin L Puterman, Maurice Queyranne. 2008. Dynamic multipriority patient scheduling for a diagnostic resource. *Operations research* **56**(6) 1507–1525.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. 2011a. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12** 2825–2830.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011b. Scikit-learn: Machine learning in python. *the Journal of machine Learning research* **12** 2825–2830.

Powell, Warren B. 2011. *Approximate Dynamic Programming: Solving the curses of dimensionality*. 2nd ed. John Wiley & Sons.

Roth, Alvin E, Tayfun Sönmez, et al. 2005. A kidney exchange clearinghouse in new england. *American Economic Review* **95**(2) 376–380.

Rothvoß, Thomas. 2017. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)* **64**(6) 1–19.

Rusmevichientong, Paat, Zuo-Jun Max Shen, David B Shmoys. 2010. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations research* **58**(6) 1666–1680.

Rusmevichientong, Paat, David Shmoys, Chaoxu Tong, Huseyin Topaloglu. 2014. Assortment optimization under the multinomial logit model with random choice parameters. *Production and Operations Management* **23**(11) 2023–2039.

Saidman, Susan L, Alvin E Roth, Tayfun Sönmez, M Utku Ünver, Francis L Delmonico. 2006. Increasing the opportunity of live kidney donation by matching for two-and three-way exchanges. *Transplantation* **81**(5) 773–782.

Samiedaluie, Saied, Beste Kucukyazici, Vedat Verter, Dan Zhang. 2017. Managing patient admissions in a neurology ward. *Operations Research* **65**(3) 635–656.

Sampson, Scott E. 2008. Or practice—optimization of vacation timeshare scheduling. *Operations research* **56**(5) 1079–1088.

Schrijver, Alexander, et al. 2003. *Combinatorial optimization: polyhedra and efficiency*, vol. 24. Springer.

Schweitzer, Paul J, Abraham Seidmann. 1985. Generalized polynomial approximations in markovian decision processes. *Journal of mathematical analysis and applications* **110**(2) 568–582.

Spivey, Michael Z, Warren B Powell. 2004. The dynamic assignment problem. *Transportation Science* **38**(4) 399–419.

StarCraftWiki. 2014. Automated matchmaking. URL `https://starcraft.fandom.com/wiki/Automated_matchmaking`.

Su, Xuanming, Stefanos A Zenios. 2005. Patient choice in kidney allocation: A sequential stochastic assignment model. *Operations research* **53**(3) 443–455.

Talluri, Kalyan, Garrett Van Ryzin. 2004. Revenue management under a general discrete choice model of consumer behavior. *Management Science* **50**(1) 15–33.

Todd, Michael J. 1977. Union jack triangulations. *Fixed Points*. Elsevier, 315–336.

Tong, Chaoxu, Huseyin Topaloglu. 2013. On the approximate linear programming approach for network revenue management problems. *INFORMS Journal on Computing* **26**(1) 121–134.

Topaloglu, H. 2009. Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management **57**(3) 637–649.

Train, K. E. 2003. *Discrete Choice Methods with Simulation*. Cambridge University Press.

Trick, Michael A, Stanley E Zin. 1997. Spline approximations to value functions: linear programming approach. *Macroeconomic Dynamics* **1**(1) 255–277.

Uber. 2019. Gaining insights in a simulated marketplace with machine learning at uber. URL `https://eng.uber.com/simulated-marketplace/`.

van Dongen, Joost. 2018. The awesomenauts matchmaking algorithm: Scoring the quality of a match. URL `http://joostdevblog.blogspot.com/2018/09/the-awesomenauts-matchmaking-algorithm.html`.

Van Ryzin, Garrett, Gustavo Vulcano. 2015. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science* **61**(2) 281–300.

van Ryzin, Garrett, Gustavo Vulcano. 2017. An expectation-maximization method to estimate a rank-based choice model of demand. *Operations Research* **65**(2) 396–407.

Vazirani, Vijay V. 2020. An extension of the birkhoff-von neumann theorem to non-bipartite graphs. *arXiv preprint arXiv:2010.05984* .

Véron, Maxime, Olivier Marin, Sébastien Monnet. 2014. Matchmaking in multi-player on-line games: studying user traces to improve the user experience. *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. ACM, 7.

Vielma, Juan Pablo, Shabbir Ahmed, George Nemhauser. 2010. Mixed-integer models for nonseparable piecewise-linear optimization: Unifying framework and extensions. *Operations research* **58**(2) 303–315.

Vielma, Juan Pablo, George L Nemhauser. 2011. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming* **128**(1) 49–72.

Virtanen, Pauli, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods* **17**(3) 261–272.

Von Luxburg, Ulrike. 2007. A tutorial on spectral clustering. *Statistics and computing* **17**(4) 395–416.

Vossen, Thomas W.M., Fan You, Dan Zhang. 2022. Finite-horizon approximate linear programs for capacity allocation over a rolling horizon. *Production and Operations Management* .

Vossen, Thomas W.M., Dan Zhang. 2015. Reductions of approximate linear programs for network revenue management. *Operations Research* **63**(6) 1352–1371.

Vulcano, Gustavo, Garrett Van Ryzin, Wassim Chaar. 2010. Om practice—choice-based revenue management: An empirical study of estimation and optimization. *Manufacturing & Service Operations Management* **12**(3) 371–392.

Wang, Xue, Mike Mingcheng Wei, Tao Yao. 2019. Online assortment optimization with high-dimensional data. *Available at SSRN 3521843* .

Wang, Yu, Aradhna Krishna. 2006. Timeshare exchange mechanisms. *Management Science* **52**(8) 1223–1237.

Wedel, Michel, Wagner A Kamakura. 2012. *Market Segmentation: Conceptual and Methodological Foundations*, vol. 8. Springer Science & Business Media.

Williams, Huw CWL. 1977. On the formation of travel demand models and economic evaluation measures of user benefit. *Environment and planning A* **9**(3) 285–344.

Xu, Mingkuan, Yang Yu, Chenye Wu. 2019. Rule designs for optimal online game matchmaking. *arXiv preprint arXiv:1911.11852* .

Yan, Chiwei, Helin Zhu, Nikita Korolko, Dawn Woodward. 2019. Dynamic pricing and matching in ride-hailing platforms. *Forthcoming, Naval Research Logistics* .

Yunes, Tallys H, Dominic Napolitano, Alan Scheller-Wolf, Sridhar Tayur. 2007. Building efficient product portfolios at john deere and company. *Operations Research* **55**(4) 615–629.

Zhang, Dan, Daniel Adelman. 2009. An approximate dynamic programming approach to network revenue management with customer choice. *Transportation Science* **43**(3) 381–394.

Zhang, Dan, William L Cooper. 2005. Revenue management for parallel flights with customer-choice behavior. *Operations Research* **53**(3) 415–431.

Zhang, Heng, Paat Rusmevichientong, Huseyin Topaloglu. 2020. Assortment optimization under the paired combinatorial logit model. *Operations Research* **68**(3) 741–761.

Zhang, Lingyu, Tao Hu, Yue Min, Guobin Wu, Junying Zhang, Pengcheng Feng, Pinghua Gong, Jieping Ye. 2017. A taxi order dispatch model based on combinatorial optimization. *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 2151–2159.

Zhong, Y, Z Wan, ZJM Shen. 2019. Balancing supply and demand: Queuing vs. surge pricing mechanisms. Tech. rep., Working paper, University of Chicago, IL.

# Appendix A

# Appendix to Chapter 2

## A.1. Technical Proofs

### A.1.1 Proof of Proposition 2.1

The proof proceeds in two steps. First, we show that ($\mathbf{RD}^A$) is a relaxation of the approximate LP dual ($\mathbf{D}^A$), that is, for any solution of ($\mathbf{D}^A$) we can construct a feasible solution with the same objective value to ($\mathbf{RD}^A$) by aggregating variables. Second, we establish the reverse direction by showing that ($\mathbf{D}^A$) can be interpreted as a Dantzig-Wolfe reformulation of ($\mathbf{RD}^A$).

To establish the first step, consider a feasible solution $\boldsymbol{\pi}$ to ($\mathbf{D}^A$) and define the following variable aggregation:

$$\hat{s}_{t,i} = \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} s_i \pi_t(\mathbf{s},\mathbf{x}), \quad \forall t \in \mathcal{T}, i \in \mathcal{V}, \tag{A.1}$$

$$\hat{x}_{t,c} = \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} x_c \pi_t(\mathbf{s},\mathbf{x}), \quad \forall t \in \mathcal{T}, c \in \mathcal{C}(L). \tag{A.2}$$

By construction, the solution $(\hat{\mathbf{s}}, \hat{\mathbf{x}})$ will satisfy constraints (9) and (11) in ($\mathbf{RD}^A$), and have the same objective function value. Constraint (10) is satisfied because for all $t \in \mathcal{T}$ and $i \in \mathcal{V}$, we have that

$$\hat{s}_{t,i} - \sum_{c\in\mathcal{C}(L), i\in c} \hat{x}_{t,c} = \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} s_i \pi_t(\mathbf{s},\mathbf{x}) - \sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \sum_{(\mathbf{s},\mathbf{x})\in X} x_c \pi_t(\mathbf{s},\mathbf{x}) = \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}} \Big(s_i - \sum_{\substack{c\in\mathcal{C}(L):\\ i\in c}} x_c\Big)\pi_t(\mathbf{s},\mathbf{a}) \geq 0.$$

To show the reverse direction, we first define $\Omega_t$ to be the set of all feasible solutions to ($\mathbf{CG\text{-}LP}_t$). Observe that $\Omega_t$ is bounded and has integer extreme points for all $t$. Let $\mathcal{X}_t^\Omega$ be the set of extreme points of $\Omega_t$, and observe that $\mathcal{X}_t^\Omega \subseteq \Omega_t$ by construction.

Now, consider a feasible solution $\{(\hat{\mathbf{s}}, \hat{\mathbf{x}})\}_{t\in\mathcal{T}}$ to $(\mathbf{RD}^A)$. For each $t \in \mathcal{T}$, we can express the solution as a convex combination of all extreme points in $\Omega_t$,

$$(\hat{\mathbf{s}}_t, \hat{\mathbf{x}}_t) = \sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}_t^\Omega} (\mathbf{s}, \mathbf{x})\mu_t(\mathbf{s}, \mathbf{x}),$$

with $\mu_t(\mathbf{s}, \mathbf{x}) \geq 0$ for all $(\mathbf{s}, \mathbf{x}) \in \mathcal{X}_t^\Omega$ and $\sum_{(\mathbf{s},\mathbf{x})\in\mathcal{X}_t^\Omega} \mu_t(\mathbf{s}, \mathbf{x}) = 1$. Therefore, we are able to construct a solution to the original ALP dual $(\mathbf{D}^A)$ by defining

$$\pi_t(\mathbf{s}, \mathbf{x}) = \begin{cases} \mu_t(\mathbf{s}, \mathbf{x}), & \text{if } (\mathbf{s}, \mathbf{x}) \in \mathcal{X}_t^\Omega, \\ 0, & \text{otherwise}, \end{cases} \qquad \forall (\mathbf{s}, \mathbf{x}) \in \mathcal{X}, t \in \mathcal{T}.$$

The resulting solution satisfies the constraints in $(\mathbf{D}^A)$ and has the same objective value as the corresponding solution to $(\mathbf{RD}^A)$. This completes the proof.

### A.1.2 Proof of Proposition 2.2

As a first step, we observe that cycles of length 2 in the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ can be interpreted as edges in a corresponding undirected graph. As a result the column generation subproblems $(\mathbf{CG}_t)$ define a $b$-matching problem, and Edmonds (1965) shows that the convex hull of $b$-matchings can be expressed using the so-called **blossom inequalities**.

When $t = 1$, we can therefore express the linear programming formulation $(\mathbf{CG\text{-}LP}_t)$ as

$$\begin{aligned} \max_{\hat{\mathbf{x}}} \quad & \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c \left( r_{t,c} - \sum_{i\in\mathcal{V}(c)} \overline{\nu}_i V_{t+1,i} \right) \hat{x}_c - \sum_{i\in\mathcal{V}} (V_{t,i} - \overline{\nu}_i V_{t+1,i} + w_{t,i}) \, s_{1,i} + \sum_{i\in\mathcal{V}} \lambda_i V_{t+1,i} - \theta_t + \theta_{t+1} \\ \text{s.t.} \quad & \sum_{c\in\mathcal{C}(L): i\in c} \hat{x}_c \leq s_{1,i}, && \forall i \in \mathcal{V}, \\ & \sum_{\substack{c\in\mathcal{C}(L): \\ \mathcal{V}(c)\in\mathcal{W}}} \hat{x}_c \leq \frac{\sum_{i\in\mathcal{W}} s_{1,i} - 1}{2}, && \forall \mathcal{W} \subseteq \mathcal{V} \text{ with } \sum_{i\in\mathcal{W}} s_{1,i} \text{ odd}, \qquad \text{(A.3)} \\ & \hat{x}_c \geq 0, && \forall c \in \mathcal{C}(L). \end{aligned}$$

Constraint (A.3) defines the blossom inequalities, which are defined over all subsets $\mathcal{W}$ of agent types such that the total number of agents (with an agent type in $\mathcal{W}$) present in the system is odd. When $t \geq 2$, we can use the variable substitution $\hat{y}_i = S - \hat{s}_i$ to express ($\mathbf{CG\text{-}LP}_t$) as

$$\begin{aligned}
\max_{\hat{\mathbf{x}}} \quad & \sum_{c \in \mathcal{C}(L)} \overline{\mu}_c \left( r_{t,c} - \sum_{i \in \mathcal{V}(c)} \overline{\nu}_i V_{t+1,i} \right) \hat{x}_c - \sum_{i \in \mathcal{V}} \left( V_{t,i} - \overline{\nu}_i V_{t+1,i} + w_{t,i} \right) (S - \hat{y}_i) + \sum_{i \in \mathcal{V}} \lambda_i V_{t+1,i} - \theta_t + \theta_{t+1} \\
\text{s.t.} \quad & \sum_{c \in \mathcal{C}(L): i \in c} \hat{x}_c + \hat{y}_i = S, \qquad \forall i \in \mathcal{V}, \\
& \sum_{\substack{c \in \mathcal{C}(L): \\ \mathcal{V}(c) \in \mathcal{W}}} \hat{x}_c \leq \frac{|\mathcal{W}|S - 1}{2}, \qquad \forall \mathcal{W} \subseteq \mathcal{V} \text{ with } |\mathcal{W}|S \text{ odd}, \qquad \text{(A.4)} \\
& \hat{x}_c \geq 0, \qquad \forall c \in \mathcal{C}(L), \\
& \hat{y}_i \geq 0, \qquad \forall i \in \mathcal{V}.
\end{aligned}$$

We observe the blossom inequalities (A.4) do not involve the slack variables $y_i$. While we can interpret these variables as edges between node $i$ and a "dummy" node, blossom inequalities over sets that include this dummy node would be redundant.

The proof follows by substituting constraints (8) in ($\mathbf{RD}^A$) by constraints (15) when $t = 1$, and removing constraints (8) when $t \geq 2$. The latter follows by observing that constraints (A.4) are redundant if we choose $S$ to be an sufficiently large and **even** integer.

### A.1.3 Proof of Proposition 2.3

Let $\mathbf{x}^* \in \mathbb{R}_+^{\mathcal{C}(L)}$ be a (possibly fractional) first period solution of the ALP reformulation, given an initial state $\mathbf{s}$. We assume $L = 2$ and therefore cycles correspond to edges $\{i, j\}$ in an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Replacing cycles by the corresponding edges and using $\delta(i)$ to represents the *family* of edges (that is, self-loops occur twice) incident on $i$, we therefore have:

$$\sum_{e\in\delta(i)} x^*_e \leq s_i, \qquad \forall i \in \mathcal{V}, \tag{A.5}$$

$$\sum_{e\in\mathcal{E}(\mathcal{W})} x^*_e \leq \frac{\sum\limits_{i\in\mathcal{W}} s_i - 1}{2}, \qquad \forall \mathcal{W} \subseteq \mathcal{V} : \sum_{i\in\mathcal{W}} s_i \text{ odd}, \tag{A.6}$$

$$\hat{\mathbf{x}} \geq 0.$$

We allow $\mathcal{E}$ to include self-loops. Observe that constraint (A.6) implies that $x_e = 0$ if $s_i = 1$ for any self loop at $i$.

As a first step, we convert this solution to a fractional **perfect** b-matching instance by adding a dummy node 0. Specifically, we let $\mathcal{V}^+ = \mathcal{V} \cup \{0\}$, $\mathcal{E}^+ = \mathcal{E} \cup \{\{0,i\} : i \in \mathcal{V}\} \cup \{0,0\}$, and let

$$s_0 = \sum_{i\in V} s_i, \text{ and}$$

$$x^*_{\{0,i\}} = s_i - \sum_{e\in\delta(i)} x^*_e, \text{ and}$$

$$x^*_{\{0,0\}} = \frac{s_0 - \sum_{i\in V} x^*_{\{0,i\}}}{2} = \sum_{e\in\mathcal{E}} x^*_e.$$

We establish that this solution satisfies constraints (A.5) and (A.6) defined on the extended graph. Constraint (A.5) will hold by construction, as will constraint (A.6) when $0 \notin \mathcal{W}$. Now, consider a subset $\mathcal{W}^+ = \{0\} \cup \mathcal{W} \subseteq \mathcal{V}^+$ such that $\sum_{i\in\mathcal{W}^+} s_i$ is odd. Observe that this implies that $\sum_{i\in\mathcal{V}\setminus\mathcal{W}} s_i$ is also odd. Then, we need to show that the following constraint is satisfied:

$$\sum_{e\in\mathcal{E}(\mathcal{W})} x^*_e + \sum_{i\in\mathcal{W}} x^*_{\{0,i\}} + x^*_{\{0,0\}} \leq \frac{s_0 + \sum_{i\in\mathcal{W}} s_i - 1}{2} \tag{A.7}$$

We first observe that

$$\begin{aligned}\sum_{i\in\mathcal{W}} x^*_{\{0,i\}} &= \sum_{i\in\mathcal{W}} s_i - \sum_{i\in\mathcal{W}} \sum_{e\in\delta(i)} x^*_e \\ &= \sum_{i\in\mathcal{W}} s_i - 2\sum_{e\in\mathcal{E}(\mathcal{W})} x^*_e - \sum_{e\in\delta(\mathcal{W})} x^*_e.\end{aligned}$$

Plugging this into constraint (A.7) yields

$$x^*_{\{0,0\}} - \sum_{e \in \mathcal{E}(\mathcal{W})} x^*_e - \sum_{e \in \delta(\mathcal{W})} x^*_e \leq \frac{\sum_{i \in \mathcal{V} \setminus \mathcal{W}} s_i - 1}{2}.$$

Because $x^*_{\{0,0\}} = \sum_{e \in \mathcal{E}} x^*_e$, this reduces to

$$\sum_{e \in \mathcal{V} \setminus \mathcal{W}} x^*_e \leq \frac{\sum_{i \in \mathcal{V} \setminus \mathcal{W}} s_i - 1}{2},$$

which hold as it is a blossom inequality present in the original constraints.

Next, we transform this fractional perfect **b-matching** solution into a fractional solution for a perfect matching problem, using the construction outlined in Schrijver et al. (2003, p. 547). Consider a graph $(\mathcal{V}_b^+, \mathcal{E}_b^+)$ obtained by splitting each node into $s_i$ copies. Formally, let $\mathcal{V}_b^+ := \{i_b \mid \forall i \in \mathcal{V}^+, 1 \leq b \leq s_i\}$, and $\mathcal{E}_b^+ := \{\{i_b, i'_{b'}\} \mid \forall \{i, i'\} \in \mathcal{E}^+, 1 \leq b \leq s_i, 1 \leq b' \leq s_{i'}, i_b \neq i'_{b'}\}$. For any edge $\{i, j\} \in \mathcal{E}_b^+$, with $i$ and $j$ copies of $i'$ and $j'$ in $(\mathcal{V}^+, \mathcal{E}^+)$, let $e' = \{i', j'\}$, define $x'_e = x^*_{e'}/(s_{i'} s_{j'})$ when $i' \neq j'$, and $x'_e = x^*_{e'}/(s_{i'}(s_{i'} - 1)/2)$ when $i' = j'$. We show that $\mathbf{x}'$ satisfy

$$\sum_{e \in \delta_b(i)} x'_e \leq 1, \qquad \forall i \in \mathcal{V}_b^+, \tag{A.8}$$

$$\sum_{e \in \mathcal{E}_b^+(\mathcal{W})} x'_e \leq \lfloor |\mathcal{W}|/2 \rfloor, \qquad \forall \mathcal{W} \subseteq \mathcal{V}_b^+ : |\mathcal{W}| \text{ odd}, \tag{A.9}$$

$$\hat{\mathbf{x}} \geq 0,$$

where $\delta_b(i)$ denotes edges incident to $i$ on $(\mathcal{V}_b^+, \mathcal{E}_b^+)$, and $\mathcal{E}_b^+(\mathcal{W})$ is the set of edges with both nodes in $\mathcal{W}$ on $(\mathcal{V}_b^+, \mathcal{E}_b^+)$ including self-loops.

To see (A.8), suppose $i \in \mathcal{V}_b^+$ is a copy of $i'$ on $(\mathcal{V}^+, \mathcal{E}^+)$. Then

$$\begin{aligned}
\sum_{e \in \delta_b(i)} x'_e &= (s_{i'} - 1) \frac{2x^*_{\{i',i'\}}}{s_{i'}(s_{i'} - 1)} + \sum_{e \in \delta(i') \setminus \{i',i'\}} \frac{x^*_e}{s_{i'}} \\
&= \frac{1}{s_{i'}} (2x^*_{\{i',i'\}} + \sum_{e \in \delta(i') \setminus \{i',i'\}} x^*_e) \\
&= \frac{1}{s_{i'}} \sum_{e \in \delta(i')} x^*_e \leq \frac{1}{s_{i'}} s_{i'} = 1.
\end{aligned}$$

For any node $v \in \mathcal{V}^+$, let $\mathcal{B}_v$ denote the set of copies of $v$ in $\mathcal{V}_b^+$. To see (A.9), let $\mathcal{W}' \subset \mathcal{V}_b^+$ with $|\mathcal{W}'|$ odd. Note that if $\mathcal{B}_v \subseteq \mathcal{W}'$ for all $v \in \mathcal{W}'$, (A.9) follows directly from (A.7). That is, we are only concerned with cases where $\mathcal{W}'$ "splits" $\mathcal{B}_v$. Without loss of generality, let $v \in \mathcal{V}^+$ be the only node such that $\mathcal{B}_v \cap \mathcal{W}' \neq \emptyset$ and $\mathcal{B}_v \not\subseteq \mathcal{W}'$. Denote $b := |\mathcal{B}_v|$ and $d := |\mathcal{B}_v \cap \mathcal{W}'|$, it follows that $0 < d < b$. Also, let $\mathcal{U}_1 := \mathcal{W}' \setminus \mathcal{B}_v$ and $\mathcal{U}_2 := \mathcal{W}' \cup \mathcal{B}_v$; it follows that $\mathcal{U}_2 \setminus \mathcal{U}_1 = \mathcal{B}_v$ and $|\mathcal{U}_2| - |\mathcal{U}_1| = b$.

Let $l$ be the edge weight of self-loops in $\mathcal{B}_v$, and $y := \sum_{u \in \mathcal{U}_1} x'_{\{v',u\}}, \forall v' \in \mathcal{B}_v$ is the sum of edge weights connecting a copy of $v$ and nodes in $\mathcal{U}_1$. For notational brevity, let $x_1 := \sum_{e \in \mathcal{E}_b^+(\mathcal{U}_1)} x'_e$. With the above definitions, because $\mathcal{U}_1$ and $\mathcal{U}_2$ do not split $v$, from (A.7) and (A.8) we have the following

$$\begin{aligned}
&y + (b-1)l \leq 1,\\
&2x_1 + b \cdot y \leq |\mathcal{U}_1|,\\
&x_1 \leq \lfloor |\mathcal{U}_1|/2 \rfloor,\\
&x_1 + b \cdot y + \frac{b(b-1)}{2} l \leq \lfloor (|\mathcal{U}_1| + b)/2 \rfloor.
\end{aligned}$$

If $b \geq 2d$, we have

$$\begin{aligned}
\sum_{e \in \mathcal{E}_b^+(\mathcal{W}')} x'_e &= x_1 + d \cdot y + \frac{d(d-1)l}{2}\\
&= \frac{b-2d}{b} x_1 + \frac{d}{b}(2x_1 + b \cdot y) + \frac{d(d-1)l}{2}\\
&\leq \frac{(b-2d)|\mathcal{U}_1|}{2b} + \frac{d|\mathcal{U}_1|}{b} + \frac{d-1}{2}\\
&= \frac{|\mathcal{U}_1| + d - 1}{2}\\
&= \lfloor |\mathcal{W}'|/2 \rfloor.
\end{aligned}$$

If $b < 2d$, it holds that

$$\begin{aligned}
\sum_{e \in \mathcal{E}_b^+(\mathcal{W}')} x'_e &= x_1 + d \cdot y + \frac{d(d-1)l}{2} \\
&= \frac{2d-b}{b}\left(x_1 + b \cdot y + \frac{b(b-1)l}{2}\right) + \frac{b-d}{b}(2x_1 + b \cdot y) + \frac{d(d-1)l}{2} - \frac{(2d-b)(b-1)l}{2} \\
&\leq \frac{2d-b}{b} \cdot \frac{|\mathcal{U}_1| + b}{2} + \frac{b-d}{b}|\mathcal{U}_1| + \frac{(b-d)(b-d-1)l}{2} \\
&= \frac{|\mathcal{U}_1| + d - 1}{2} - \frac{b-d-1}{2} + \frac{(b-d)(b-d-1)l}{2} \\
&= \frac{|\mathcal{U}_1| + d - 1}{2} + \frac{b-d-1}{2}\left((b-d)l - 1\right) \\
&\leq \frac{|\mathcal{U}_1| + d - 1}{2} \\
&= \lfloor |\mathcal{W}'|/2 \rfloor.
\end{aligned}$$

As a next step, we can proceed to show (A.9) by induction on the number of nodes "split" by $\mathcal{W}'$. Note that any points of the polytope defined by (A.8) and (A.9) are also feasible for (A.5) and (A.6) by construction.

In Vazirani (2020), the author show that a fractional non-bipartite perfect matching can be written as a convex combination of perfect matchings, extending the Birkhoff-von Neumann Theorem from bipartite to non-bipartite graphs. Applying the algorithm proposed in Vazirani (2020) to $\mathbf{x}'$ on $(\mathcal{V}_b^+, \mathcal{E}_b^+)$, we obtain a convex combination of perfect matchings, each with a corresponding feasible matching on the original compatibility graph; this completes the proof.

## A.2. Connection to Other Applications

### A.2.1 Network Revenue Management

Consider the following network revenue management (NRM) problem, where we use airline terminology for ease of exposition. We define a set of flight legs as $\mathcal{I} = \{1, ..., I\}$, and define the capacity of leg $i$ as $c_i$. We also define a set of products $\mathcal{J} = \{1, ..., J\}$ and define the fare for product $j$ as $f_j$. We further have a consumption matrix $\boldsymbol{A} \equiv (a_{ij})$ of dimension $(I \times J)$, where $a_{ij} \in \{0, 1\}$ indicates whether product $j$ uses leg $i$. Time periods belong to the set $\mathcal{T} = \{1, ..., T\}$. In period $t$, the arrivals of type $j$ product demands follow Poisson process with rate $\lambda_{t,j}$, that is, we assume an independent demand model

To formulate the NRM problem with batch demand as an MDP, we define the state space as $(\mathbf{s}^l, \mathbf{s}^p)$ where $\mathbf{s}^l = \{s_1^l, ..., s_I^l\}$ represents the remaining capacity on each flight leg and $\mathbf{s}^p = \{s_1^p, ..., s_J^p\}$ corresponds to the number of requests for each product. Without loss of generality, we let $S$ be an upper bound on the number of product requests. We use $\mathcal{S}$ to represents the set of all feasible states.

Given a state $(\mathbf{s}_t^l, \mathbf{s}_t^p)$ at the start of period $t$, the planner decides which product requests to accept. For each $j \in \mathcal{J}$, we introduce variable $x_j$ to represent the number of type $j$ requests to accept. The actions are subject to capacity constraints, and therefore the action space can be defined as

$$\mathcal{A}(\mathbf{s}^l, \mathbf{s}^p) = \left\{ \mathbf{x} \in \mathbb{Z}_+^J : x_j \le s_j^p, \forall j \in \mathcal{J}, \sum_{j \in \mathcal{J}} a_{ij} x_j \le s_i^l, \forall i \in \mathcal{I} \right\}.$$

In addition, we also define

$$\mathcal{X} = \left\{ (\mathbf{s}^j, \mathbf{s}^p, \mathbf{x}) : (\mathbf{s}^j, \mathbf{s}^p) \in \mathcal{S}, \mathbf{x} \in \mathcal{A}(\mathbf{s}^l, \mathbf{s}^p) \right\}$$

to represent the set of all state-action pairs.

The system dynamics of the NRM problem with "block" demands can be described as follows. The arrival rates of flight legs are 0, and the departure probabilities of flight legs also equal 0. The arrivals of product demands follow independent Poisson processes, while unmet demands are lost; that is, the departure probabilities of demands are equal to 1. There are no match failures, no waiting costs, and the reward for $x_j$ is $f_j$ for all $j \in \mathcal{J}$.

Given the above definitions, we are ready apply the framework proposed in this chapter. Specifically, after writing down the optimality equations, applying the affine functional approximation and simplifying, we obtain the following formulation analogous to $(\mathbf{P}^A)$:

$$\begin{aligned}
\min \quad & \theta_1 + \sum_{i \in \mathcal{I}} V^l_{1,i} s^l_{1,i} + \sum_{j \in \mathcal{J}} V^p_{1,j} s^p_{1,j} \\
\text{s.t.} \quad & \theta_t + \sum_{i \in \mathcal{I}} V^l_{t,i} s^l_{t,i} + \sum_{j \in \mathcal{J}} V^p_{t,j} s^p_{t,j} \geq \theta_{t+1} + \sum_{i \in \mathcal{I}} V^l_{t+1,i} (s^l_{t,i} - \sum_{j \in \mathcal{J}} a_{ij} x_{t,j}) \\
& \qquad + \sum_{j \in \mathcal{J}} V^p_{t+1,i} \lambda_{t,j} + \sum_{j \in \mathcal{J}} f_j x_{t,j}, \quad \forall t \in \mathcal{T}, (\mathbf{s}^l, \mathbf{s}^p, \mathbf{x}) \in \mathcal{X}.
\end{aligned}$$

Now, consider the subproblems that arise when the dual of the above formulation is solved with a column generation method and suppose that solving the restricted master problem yields a dual solution $(\theta, V^l, V^p)$. For each period $t \in \mathcal{T}$, the column generation subproblem can be expressed by the following integer program

$$\begin{aligned}
\max \quad & \sum_{j \in \mathcal{J}} (f_j - \sum_{i \in \mathcal{I}} a_{ij} V^l_{t+1,i}) x_{t,j} - \sum_{i \in \mathcal{I}} V^l_{t,i} s^l_{t,i} - \sum_{j \in \mathcal{J}} V^p_{t,j} + \sum_{j \in \mathcal{J}} V^p_{t+1,i} \lambda_{t,j} + \theta_{t+1} - \theta_t \\
\text{s.t.} \quad & x_{t,j} \leq s^p_{t,j}, \quad \forall j \in \mathcal{J}, \\
& \sum_j a_{ij} x_{t,j} \leq s^l_{t,j}, \quad \forall i \in \mathcal{I}, \\
& \mathbf{s}^l_t, \mathbf{s}^p_t, \mathbf{x}_t \geq 0, \text{ integer.}
\end{aligned}$$

With block demands, this column generation subproblem corresponds to a generalized packing problem, which is difficult to solve and for which it is difficult to explicitly characterize the convex hull constraints.

However, if we restrict arrivals such that at most request arrives in each period, the column generation subproblem will in fact become easier to solve, given the additional constraint

$$\sum_{j \in \mathcal{J}} s^p_{t,j} \leq 1.$$

While we omit a formal proof of the integrality of the resulting polyhedron, we do note that the resulting problems are easy to solve. For each $j \in \mathcal{J}$, suppose $j$ is the one arriving product request. We only need to check if accepting this demand yields a higher objective value than rejecting it, which takes linear time; after going through all $J$ potential requests, we obtain the solution of the period $t$ subproblem.

### A.2.2 Dynamic Timeshare Exchange Problems

Our approach can be applied to model timeshare exchanges, whose members can exchange timeshare "weeks" they own according to their preferences; see Wang and Krishna (2006), Sampson (2008). For ease of exposition, we first introduce a static version of this problem. Let $\mathcal{V} = \{1, ..., V\}$ be the set of all types of timeshare owners. Owners of the same type not only possess week-long vacations at the same resort in the same time of the year, but also share the same preferences towards vacation weeks owned by members of a different type.

Following the model proposed in this chapter directly, we create a compatibility graph by adding a node for each $i \in \mathcal{V}$. For each type $i$ owner, if they prefer a week owned by type $j \in \mathcal{V}$ over their own own week represented by the type $i$, a directed edge is added to the graph from $i$ to $j$. For each directed edge, a positive weight is assigned to represent the relative preferences among weeks.

To represent the action space, let $\mathcal{C}$ be the set of all elementary cycles (without cycle length limit). For each $c \in \mathcal{C}$, a decision variable $x_c$ represents the number of times a cycle $c$ is scheduled for exchange. The size of $\mathcal{C}$ can be prohibitively large, and as a result the timeshare exchange problem may appear to be intractable.

However, we can define a bipartite graph $H$ to demonstrate that the problem is in fact tractable. First, for each $i \in \mathcal{V}$, we add $i$ and $i'$ to $H$ to reflect the owner and her week; we also add an edge $(i, i')$ to $H$, which means owners can always be assigned to their own week. Next, we add an arc $(i, j)$ to $H$ for each owner type $i$ if the owner prefers a week owned by type $j \in \mathcal{V}$ over their own week. Finally, we assign appropriate positive edge weights to $H$ where the weight of $(i, i')$ is 0 for all $i$. With this construction, the static timeshare exchange problem is converted to a bipartite b-matching problem, where all capacity constraints are equality constrains to ensure that every owner is assigned a week at least as good as her own. Since bipartite b-matching polytopes are integral, the problem can be solved efficiently. A trivial feasible solution of this problem is obtained by assigning every owner their own weeks.

We next discuss how the bipartite formulation could model the dynamic version, where owners of each type arrive to the timeshare exchange randomly. First, it is easy to see that waiting costs as well as arrivals and departures of owners can be modeled directly. Next, we propose two approaches to address failures after matching decisions are made: 1) suppose a matching decision at the end of a period is made and one type $i$ owner who is assigned to week $j$ is unable to go, this owner simply loses her own week and all successful matchings go as planned; 2) after one type $i$ owner who is assigned to week $j$ rejects the assignment, a trading cycle can be detected through backtracking, and every owner of the cycle return to there respective weeks. With such modifications, our approach can be applied to solve dynamic timeshare exchange problems.

## A.3. Implementation Details

As discussed in Section 4.3, we obtain an upper bound by solving $(\mathbf{RD})^A$ for the initial state $\mathbf{s}_1$. When $L = 2$, we use a constraint generation procedure to solve $(\mathbf{RD})^A$. To obtain a lower bound, we use Monte Carlo simulation. Starting from an initial state, we select actions according to the primal or dual policies described in Section 4.4 and advance the state by sampling from the stochastic process that describes the uncertain state transitions. At each state thus visited, we resolve the ALP and use the heuristic policies to generate an action. In Sections A.3.1 and A.3.2 we outline the algorithms that we use to implement our policies. Sections A.3.3 and A.3.4 describe our implementation of the benchmark policies we use.

### A.3.1 ALP Primal Policy

The primal policy we use is based on a solution to the primal program $(\mathbf{RP})^A$ that corresponds to $(\mathbf{RD})^A$. The algorithm we use to select actions according to this policy is as follows:

The ALP primal policy can be executed with or without resolving. Without resolving, Step 1 is omitted and the initial optimal solution $(\boldsymbol{V}^*, \boldsymbol{W}^*, \boldsymbol{U}^*)$ is used throughout. When $L \geq 3$, we

**Algorithm 3** ALP Primal Policy

```
**Input:** A state $\mathbf{s}$
**Output:** A matching $\mathbf{x}^\pi \in \mathcal{A}(\mathbf{s})$
**Step 1.** Solve $(\mathbf{RP})^A$ to obtain an optimal solution $(\boldsymbol{V}^*, \boldsymbol{W}^*, \boldsymbol{U}^*)$
**Step 2.** Let
```

$$\begin{aligned} \mathbf{x}^\pi = \arg \max_{\mathbf{x} \in \mathbb{Z}^{|\mathcal{C}(L)|}} \quad & \sum_{c \in \mathcal{C}(L)} \bar{\mu}_c \left( r_{1,c} - \sum_{i \in \mathcal{V}(c)} V^*_{2,i} \bar{\nu}_i \right) x_c \\ \textbf{s.t.} \quad & \sum_{\substack{c \in \mathcal{C}(L): \\ i \in \mathcal{V}(c)}} x_c \le s_i, \quad \forall i \in \mathcal{V}. \end{aligned}$$

use an efficient heuristic (that is, we round down the LP relaxation solution) to solve large problem instances; initial experiments showed that this has little impact on performance.

### A.3.2 ALP Dual Policy

The dual policy is based on an optimal solution to the ALP reformulation $(\mathbf{RD})^A$. The algorithm we use to select actions according to this policy is as follows:

**Algorithm 4** ALP Dual (Probabilistic) Policy

```
**Input:** A state $\mathbf{s}$
**Output:** A matching $\mathbf{x}^\pi \in \mathcal{A}(\mathbf{s})$
**Step 1.** Solve $(\mathbf{RD})^A$ to obtain an optimal solution $\mathbf{x}^*$
**Step 2.** **If** $L = 2$:
            Generate $\rho_n, \mathbf{x}_1^n$ as described in Proposition 3, and let $\mathbf{x}^\pi = \mathbf{x}_1^n$ with probability $\rho_n$
        **Else**:
            $x_c^\pi = \lfloor x^*_{1,c} \rfloor, \quad \forall c \in \mathcal{C}(L)$
```

When executing the ALP dual policy, it is necessary to resolve the ALP at the beginning of each period. For instances with $L \geq 3$, we simply round down the first period solutions, and we use a probabilistic allocation method when $L = 2$. In all cases, the dual policy has polynomial time and space complexity.

### A.3.3 Deterministic Linear Program

The multi-stage deterministic linear program we use equals

$$\mathbf{DLP} : \max_{\hat{\mathbf{s}},\hat{\mathbf{x}}} \sum_{t\in\mathcal{T}} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c} \hat{x}_{t,c} - \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{V}} w_{t,i} \hat{s}_{t,i}$$

$$\text{s.t.} \quad \hat{s}_{t,i} = \begin{cases} s_i, & \text{if } t = 1; \\ \overline{\nu}_i \hat{s}_{t-1,i} - \sum_{\substack{c\in\mathcal{C}(L): \\ i\in\mathcal{V}(c)}} \overline{\mu}_c \overline{\nu}_i \hat{x}_{t-1,c} + \lambda_i, & \text{if } t > 1. \end{cases} \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}, \tag{A.10}$$

$$\hat{x}_{t,c} \geq 0, \qquad \forall t \in \mathcal{T}, c \in \mathcal{C}(L),$$

$$\hat{s}_{t,i} \geq 0, \qquad \forall t \in \mathcal{T}, i \in \mathcal{V}.$$

The DLP keeps track of state evolution in expectation while maintaining state variable feasibility, and it is easy to show that the optimal objective value of the DLP provides an upper bound.

Based on the optimal solution of the DLP, we construct a control policy that is similar to our primal policy. Specifically, we interpret the optimal dual solution associated with constraint (A.10) as marginal values of agents in the system, and for every $c \in \mathcal{C}(L)$, we only make matching $c$ available when its reward is higher than the opportunity cost of resources (agents) consumed. To account for failures and agent departures, we adjust the marginal values accordingly. The algorithm we use to select actions using this DLP policy is as follows:

**Algorithm 5** DLP Policy

```
Input: A state $\mathbf{s}$
Output: A matching $\mathbf{x}^\pi \in \mathcal{A}(\mathbf{s})$
Step 1. Solve DLP to obtain an optimal dual solution $\boldsymbol{V}^*$ associated with (A.10)
Step 2. Let
    $$\mathcal{C}' = \{c \in \mathcal{C}(L) : \bar{\mu}_c \left( r_{1,c} - \sum_{i\in\mathcal{V}(c)} V^*_{2,i}\overline{\nu}_i \right) x_c \geq 0\}$$
Step 3. Let
    $$\mathbf{x}^\pi = \arg \max_{\mathbf{x}\in\mathbb{Z}^{|\mathcal{C}'|}} \sum_{c\in\mathcal{C}'} \bar{\mu}_c \left( r_{1,c} - \sum_{i\in\mathcal{V}(c)} V^*_{2,i}\overline{\nu}_i \right) x_c$$
    $$\textbf{s.t.} \sum_{\substack{c\in\mathcal{C}': \\ i\in\mathcal{V}(c)}} x_c \leq s_i, \quad \forall i \in \mathcal{V}.$$
```

**Algorithm 6** Limited Lookahead Policy

```
**Input:** A state $\mathbf{s}$, a lookahead horizon length $H$
**Output:** A matching $\mathbf{x}^\pi \in \mathcal{A}(\mathbf{s})$
**Step 1.** Let
```

$$\mathbf{x}^* = \operatorname*{argmax}_{\hat{\mathbf{s}},\hat{\mathbf{x}}} \sum_{t=1}^{1+H} \sum_{c\in\mathcal{C}(L)} \overline{\mu}_c r_{t,c} \hat{x}_{t,c} - \sum_{t=1}^{1+H} \sum_{i\in\mathcal{V}} w_{t,i}\hat{s}_{t,i}$$

$$\begin{aligned}
\text{s.t.} \quad & \hat{s}_{t,i} = \begin{cases} s_i, & \text{if } t=1; \\ \overline{\nu}_i \hat{s}_{t-1,i} - \sum\limits_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \overline{\mu}_c \overline{\nu}_i \hat{x}_{t-1,c} + \lambda_i, & \text{if } t>1. \end{cases} && \forall t=1,\dots,1+H, i\in\mathcal{V}, \\
& \sum_{\substack{c\in\mathcal{C}(L):\\ i\in\mathcal{V}(c)}} \hat{x}_{t,c} \le \hat{s}_{t,i}, && \forall t=1,\dots,1+H, i\in\mathcal{V}, \\
& \hat{x}_{t,c} \ge 0, && \forall t=1,\dots,1+H, c\in\mathcal{C}(L), \\
& \hat{s}_{t,i} \ge 0, && \forall t=1,\dots,1+H, i\in\mathcal{V}.
\end{aligned}$$

```
**Step 2.** Let $x^\pi_c = \lfloor x^*_{1,c} \rfloor, \quad \forall c \in \mathcal{C}(L)$
```

### A.3.4 Limited Lookahead Policy

We also implement a limited lookahead policy by optimizing matching decisions over a pre-specified horizon $H$, where random values are replaced by their respective expectations, and integrality of decision variables is relaxed for tractability. Our implementation of this policy uses the following algorithm:

## A.4. Additional Numerical Experiment Details

### A.4.1 Dynamic Matching for Ridesharing

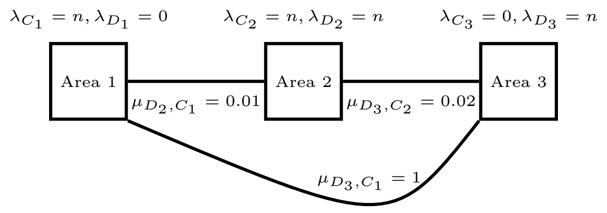
Recently, there has been considerable research on dynamic matching problems for ridesharing platforms. A common approach is to greedily match customers to drivers based on distance and create price incentives to balance supply and demand, see for example, Chen et al. (2015), Banerjee et al. (2015), and Chen and Sheldon (2016). Only a few papers study the impact of optimizing matching decisions. Hu and Zhou (2022) derive conditions under which a myopic matching policy is optimal for two-sided systems where the objective is to maximize expected total profit. In Banerjee et al. (2018) and Kanoria and Qian (2019), the authors model the two-sided, continuous-time ridesharing system as a closed queueing network where a constant number of drivers are relocated subject to matching decisions, and propose algorithms to manage the geographical distribution of supply.

Our model is closely related to a stream of work where the ridesharing system is modeled as an open queueing network, see Özkan and Ward (2020) and Aouad and Saritac (2019). In these papers, arrivals of supply and demand are independent of the matching decisions. There is support for such an assumption in the work of Zhong et al. (2019), who use data from the ride-sharing company Didi to show that a Markovian queueing model with exogenous customer and driver behaviors provides good estimates for system performance measures during rush hours in China. We first evaluate the performance of our ALP approach using an example from Özkan and Ward (2020). Next, we also evaluate our approach on randomly generated larger instances to assess the scalability of our approach.

#### A.4.1.1 Setup for Initial Experiments.

Our initial experiments consider an example from Özkan and Ward (2020), as shown in Figure A.1. In the example instance, the ridesharing region is partitioned into 3 disjoint areas. Area 1 only has customer arrivals while area 3 only has driver arrivals, causing an imbalance between supply and demand in these regions. The time horizon is 10 time units. Drivers and customers arrive to the system according to a Poisson process such that the expected number of drivers arriving at Area 1, Area 2 and Area 3 is $0, n$ and $n$ per time unit, and the expected number of customers arriving at Area 1, Area 2 and Area 3 is $n, n$ and 0 per time unit, respectively. Without loss of generality, we assume the Poisson distribution is truncated after some large number to ensure a finite state space. Here, $n$ is a parameter that represents the *thickness* of the matching market. If not matched, the probability that drivers remain in the system follows an exponential distribution with mean 10, whereas customers leave immediately. In the continuous time setting, decision epochs are the times at which agents arrive. There is no waiting cost, and the overall objective is to maximize the total expected number of successful matchings.

We construct a dynamic matching instance as shown in Figure A.2, using an embedded


$\lambda_{C_1} = n, \lambda_{D_1} = 0$
$\lambda_{C_2} = n, \lambda_{D_2} = n$
$\lambda_{C_3} = 0, \lambda_{D_3} = n$
Area 1
Area 2
Area 3
$\mu_{D_2,C_1} = 0.01$
$\mu_{D_3,C_2} = 0.02$
$\mu_{D_3,C_1} = 1$
Figure A.1: Example from Özkan and Ward (2020)

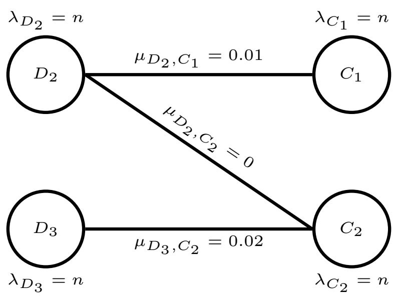


Figure A.2: Dynamic matching instance for the example from Özkan and Ward (2020)

discrete-time model such that the probability of more than one customer arriving during each period is zero. In the compatibility graph, nodes $D_2$ and $D_3$ represents drivers that arrive at areas 2 and 3, and $C_1$ and $C_2$ represent customers that arrive at areas 1 and 2. Nodes $D_2$ and $C_2$ represent drivers and customers in the same area, and therefore the probability of a successful match equals 1; similarly, the success probability of a match between $D2$ and $C1$ equals 0.99, and the success probability of a match between $D3$ and $C2$ equals 0.98, as the distances between these areas are relatively close. The success probability of a match between $D3$ and $C1$ equals 0 because we assume that areas 1 and 3 are too distant. The number of arrivals in each time unit is Poisson with rate $n$ for all nodes. We assume that $\nu_{C_1}$ and $\nu_{C_2}$ equal 1, that is, customers leave immediately when not matched; in addition, $\nu_{D2}$ and $\nu_{D_3}$ are such that the time that drivers remain in the system follows an exponential distribution with mean 10 time units. Since we are maximizing the number of matchings, the expected rewards are equal to the success rates for the matchings.

#### A.4.1.2 Setup and Results for Large Instances.

As a first step, we randomly generate networks that correspond to ridesharing regions using the Erdős–Rényi model (Erdos et al. 1960). Each node in the network represents an area in the region, while edges represent connectivity between the areas; that is, we can only match drivers and customers in areas that are connected. Furthermore, we assume that there are two driver classes, economy and premium, and two customer classes, regular and priority. Drivers and customers arrive at each area following independent Poisson processes with randomly generated rates. Again, we assume the Poisson distribution is truncated after some large number to ensure a finite state space. Match failure probabilities are based on the distance between regions (shortest path between nodes on the random graph): matches between areas with longer distances have higher failure rates. Rather than maximizing the total number of matches, we consider a more sophisticated reward structure. Specifically, matching priority customers with premium drivers yields higher reward while matching customers with economy drivers yields lower reward; regular customers and premium drivers are incompatible. Moreover, we also allow a short sojourn time for customers. There is no waiting cost. By slightly increasing potential latency of matching decisions, the platform accumulates a *batch* of customer requests, resulting in a more efficient assignment of the drivers. We assume the expected sojourn time is 15 periods for drivers, and 5 periods for customers. In each period, we assume that 20 drivers and 20 customers arrive on average. The horizon equals 100 time periods.

In our experiments, we vary the network size and randomly generate 5 instances with 5 areas, 5 instances with 10 areas, and 5 instances with 20 areas. Table A.1 reports the average computation times of the ALP reformulation, solving the ALP using column generation, and the DLP. Tables A.2 and A.3 summarize policy performance for instances with 5 and 10 areas.

Table A.1: Computation times (seconds) of larger dynamic ridesharing instances

| Size | ALP | ALP_CG | DLP |
|---|---|---|---|
| 5 | 0.24 | 1337 | 0.19 |
| 10 | 0.95 | >3600 | 0.39 |
| 20 | 3.64 | >3600 | 1.95 |

Table A.2: Upper and lower bounds for ridesharing instances with 5 areas

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 1135.1 | 851.5 | 1059.4 | 780.3 | 1008.8 |
| DLP UB | | 1168.5 | 963.4 | 1090.9 | 840.4 | 1206.4 |
| ALP Dual Policy | lb(se) | 1131.3(2.5) | 847.7(1.9) | 1049.9(2.0) | 776.3(1.7) | 1007.4(2.9) |
| | gap | 0.33 | 0.44 | 0.89 | 0.51 | 0.13 |
| ALP Primal Policy with resolve | lb(se) | 1130.0(2.5) | 846.9(2.2) | 1048.0(2.1) | 773.2(1.7) | 1006.3(2.7) |
| | gap | 0.44 | 0.54 | 1.07 | 0.91 | 0.24 |
| ALP Primal Policy no resolve | lb(se) | 1129.9(2.6) | 840.1(2.2) | 1039.4(2.2) | 771.1(1.9) | 1003.4(3.1) |
| | gap | 0.45 | 1.3 | 1.88 | 1.18 | 0.52 |
| DLP Policy | lb(se) | 959.7(2.1) | 738.0(1.6) | 854.6(3.8) | 735.0(1.9) | 775.8(2.3) |
| | gap | 17.8 | 23.3 | 21.6 | 12.5 | 35.7 |
| LLA Policy | lb(se) | 1124.5(2.6) | 842.3(2.0) | 1041.5(2.0) | 766.0(1.8) | 1001.0(2.5) |
| | gap | 0.93 | 1.07 | 1.68 | 1.83 | 0.76 |

Table A.3: Upper and lower bounds for ridesharing instances with 10 areas

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 1183.5 | 1137.7 | 1062.1 | 1060.6 | 1096.4 |
| DLP UB | | 1262.4 | 1234.6 | 1147.9 | 1110.9 | 1222.9 |
| ALP Dual Policy | lb(se) | 1165.0(2.5) | 1129.3(2.4) | 1050.6(2.2) | 1051.9(2.3) | 1087.1(2.2) |
| | gap | 1.56 | 0.73 | 1.08 | 0.82 | 0.84 |
| ALP Primal Policy with resolve | lb(se) | 1167.0(2.3) | 1128.1(2.4) | 1053.2(2.1) | 1050.9(2.4) | 1085.7(2.2) |
| | gap | 1.39 | 0.84 | 0.83 | 0.92 | 0.97 |
| ALP Primal Policy no resolve | lb(se) | 1158.9(2.4) | 1123.2(2.1) | 1040.6(2.1) | 1042.3(2.3) | 1079.6(2.4) |
| | gap | 2.07 | 1.27 | 2.02 | 1.72 | 1.52 |
| DLP Policy | lb(se) | 1018.5(2.5) | 1054.1(2.3) | 925.8(2.6) | 981.2(2.2) | 947.4(2.1) |
| | gap | 19.3 | 14.6 | 19.3 | 11.6 | 22.5 |
| LLA Policy | lb(se) | 1132.9(2.4) | 1094.9(2.1) | 1032.0(2.1) | 1026.7(2.2) | 1071.2(2.2) |
| | gap | 4.27 | 3.75 | 2.83 | 3.20 | 2.29 |

### A.4.2 Matchmaking in Online Video Games

The matchmaking system for online PvP games is a crucial service that connects players for online game sessions. A good matchmaking system can increase the players' experience and engagement, which in turn improves player retention (Butcher 2008).

A key difficulty of the matchmaking decisions in the 1v1 games we consider is balancing the quality of the match that occurs when two players are paired with the time players spend waiting for a match to begin. While the quality of a match in this setting can depend on a number of factors, two critical components are latency considerations and player skills. In matchmaking systems, players' geo-location and network latency need to be taken into account. For example, pairing players across the Pacific Ocean will inevitably result in high latency and poor player experience. To address this issue matchmaking systems often set a threshold on the latency, and player compatibility is determined based on this threshold. Predicting latency is considered by Agarwal and Lorch (2009) for console games and by Manweiler et al. (2011) for mobile games. Because players desire competitive games, and matchmaking systems aim to pair players with similar skill levels. Various models have been proposed to establish player skill levels, including Elo (Elo 1978) and TrueSkill (Herbrich et al. 2007). Typically, these models assume a prior distributional belief on each player's skill level and update the distribution's parameters after the outcome of each game session.

We assume that matchmaking systems can evaluate the quality of a pairing using a function of all relevant attributes of the pair that results in a "score". Even when the quality of a pairing can be evaluated, however, determining an optimal set of matchings presents a significant challenge. Because players join the game at different times, the quality of the matchings has to be balanced with the time players spend waiting for a game to begin. Waiting times in online video games are analyzed in Véron et al. (2014) and in Xu et al. (2019), who establish that this tradeoff can have a significant impact. This is further illustrated by the experience of Blizzard, which modified its matchmaking system for StarCraft II in April 2012 to reduce waiting times by allowing players with larger skill gaps to be paired. However, this led to an immediate outcry of complaints on social media, causing Blizzard to admit failure and introduce changes to the matchmaking system that favored more balanced and competitive games over short wait times (StarCraftWiki 2014).

#### A.4.2.1 Setup for Initial Experiments.

To create our baseline instances, we generate reward parameters $r_{t,e}$ from normal distributions with rewards as shown in Table A.4; the waiting cost parameters $w_{t,i}$ are also drawn from normal distributions shown in Table A.4. We truncate to ensure that all rewards and costs are strictly positive. The Poisson arrival rates of the player types are randomly generated such that 50% of the arrivals are mid skill players, 30% are low skill players, and 20% are high skill players. On average, a total of 4 players arrive to the matchmaking system every period. Again, we assume the Poisson distribution is truncated after some large number to ensure a finite state space. Match failures due to internet issues occur with probability 0.1% on average. The initial state for all player types is uniformly generated from the interval [0,1,2], and the time horizon is set to be 100 periods.

#### A.4.2.2 Setup and Results for Further Experiments.

To understand how our ALP approach performs in different environments, we randomly generate a number of additional instances. First, we randomly generate 5 instances where players strongly prefer fast matchmaking over fair games, by multiplying waiting costs by a factor 2. Similarly, generate 5 instances where players strongly prefer fair games over fast matchmaking, by multiplying waiting costs by a factor 0.5. In addition, we also generate 5 instances that correspond to the situation when the game is popular with a large pool of players, by multiplying the mean arrival rate by 2. Tables A.5, A.6, and A.7 summarize policy performance for these instances.

Table A.4: Matching reward and waiting cost parameters of baseline matchmaking instances

| Reward parameters | | |
|---|---|---|
| location/skill | same | adjacent |
| same | $\mathcal{N}(10, 1.00)$ | $\mathcal{N}(8, 0.64)$ |
| adjacent | $\mathcal{N}(6, 0.36)$ | $\mathcal{N}(5, 0.25)$ |

| Waiting cost parameters | | |
|---|---|---|
| high | mid | low |
| $\mathcal{N}(2, 0.04)$ | $\mathcal{N}(3, 0.09)$ | $\mathcal{N}(4, 0.16)$ |

To assess the scalability of our approach, we also generate 5 instances with a larger number of regions and skill levels. Specifically, we randomly generate instances with 8 regions and 6 skill levels. Connectivity between regions is randomly generated; in particular, internet connection quality between regions are randomly sampled from *good*, *okay* and *bad* with probability 0.5, 0.3 and 0.2, respectively. Reward structures are generated based on the network of regions, such that lower skill discrepancy and better connection quality yields higher reward. Arrival rates of each type are randomly generated such that on average a total of 100 players arrive to the system, and the departure rates are drawn such that players stay for 10 periods on average. Failure probabilities between compatible types are uniformly drawn from 0 to 2%. Table A.8 summarizes policy performance for these large size instances.

### A.4.3 Kidney Exchange

We generate instances with the commonly used *Saidman Generator* (Saidman et al. 2006), where each patient-donor pair is characterized by the attributes shown in Table A.9. This leads to 128 types of patient-donor pairs, which represent the nodes in the compatibility graph; directed edges representing medical compatibility are added according to patient and donor attributes. Similar to the approach in Li et al. (2019), the compatibility graph is weighted and edge weights are a proxy to expected quality of life after the exchange.

Table A.5: Upper and lower bounds for high waiting cost matchmaking instances

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 3133.4 | 3187.7 | 3056.7 | 2952.4 | 3146.9 |
| DLP UB | | 4370.4 | 4336.7 | 4219.2 | 3505.3 | 4484.0 |
| ALP Dual Policy | lb(se) | 2560.9(15.9) | 2524.9(17.0) | 2275.2(15.1) | 2428.5(15.8) | 2397.0(14.8) |
| | gap | 18.3 | 20.8 | 18.7 | 20.5 | 23.8 |
| ALP Primal Policy with resolve | lb(se) | 2562.8(16.5) | 2554.0(16.4) | 2351.6(14.9) | 2464.0(15.4) | 2435.9(15.3) |
| | gap | 18.2 | 19.8 | 15.9 | 19.4 | 22.6 |
| ALP Primal Policy no resolve | lb(se) | 2354.4(20.4) | 1840.9(59.9) | 2344.0(14.5) | 2343.1(17.9) | 2324.4(21.2) |
| | gap | 24.9 | 42.2 | 16.2 | 23.3 | 26.1 |
| DLP Policy | lb(se) | 2535.0(15.7) | 2514.8(15.3) | 2234.5(15.8) | 2432.3(15.8) | 2417.8(15.4) |
| | gap | 42.0 | 42.0 | 45.8 | 42.3 | 46.1 |
| LLA Policy | lb(se) | 2369.4(16.8) | 2298.1(15.8) | 2107.1(15.9) | 2246.5(17.1) | 2213.4(15.3) |
| | gap | 24.4 | 27.9 | 24.7 | 26.5 | 26.7 |

The arrivals of each patient-donor type follow independent Poisson processes, with rates determined by the attribute distributions shown in Table A.9. Following Dickerson et al. (2018), we use a time horizon of 24 periods (weeks). In each period, the probability that a patient-donor pair will leave equals 0.0035. Failure probabilities of edges are also determined by attributes, and between certain types the failure probability can be as high as 95%; the overall failure probability is over 60%.

#### A.4.3.1 Experiments with Column Generation methods.

Because solving the ALP with column generation is intractable for our original kidney exchange instances, we also generate smaller instances with cycle length restrictions $L = 3$ and $L = 4$. Using these instances, we can evaluate how the ALP bounds are impacted by the removal of constraints in our relaxed ALP (see Section 4.3 for additional discussion).

We generate random graphs with varying sizes $V$ (that is, the number of nodes in the compatibility graph) and cycle length restrictions $L$. For each $(V, L)$ combination, we create 100 instances with random rewards, arrival and departure rates, and failure probabilities. For each instance, we solve both the (relaxed) ALP reformulation and the ALP using column generation.

Table A.10 summarizes our results, and shows computation times and bounds obtained for both methods. We also report the relative difference in bounds, as well as the number of instances

Table A.6: Upper and lower bounds for low waiting cost matchmaking instances

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 4028.5 | 3767.3 | 3912.0 | 4130.4 | 4040.9 |
| DLP UB | | 4348.3 | 4129.9 | 4234.7 | 4498.2 | 4383.3 |
| ALP Dual Policy | lb(se) | 3653.3(20.7) | 3521.1(20.4) | 3637.9(20.4) | 3736.8(20.4) | 3715.9(21.0) |
| | gap | 9.3 | 6.5 | 7.0 | 9.5 | 8.0 |
| ALP Primal Policy with resolve | lb(se) | 3632.2(20.8) | 3507.3(21.0) | 3619.6(20.7) | 3732.9(20.1) | 3659.0(22.6) |
| | gap | 9.8 | 6.9 | 7.5 | 9.6 | 9.5 |
| ALP Primal Policy no resolve | lb(se) | 3247.4(32.3) | 3353.3(27.7) | 3441.9(25.7) | 2730.5(49.1) | 3416.1(29.6) |
| | gap | 19.4 | 11.0 | 12.0 | 33.9 | 15.5 |
| DLP Policy | lb(se) | 3615.1(20.4) | 3509.2(20.2) | 3627.5(20.3) | 3718.5(20.5) | 3682.2(20.8) |
| | gap | 16.9 | 15.0 | 14.3 | 17.3 | 16.0 |
| LLA Policy | lb(se) | 3586.4(21.1) | 3438.8(20.4) | 3571.2(20.3) | 3662.4(20.8) | 3657.8(21.0) |
| | gap | 11.0 | 8.7 | 8.7 | 11.3 | 9.5 |

Table A.7: Upper and lower bounds for high arrival rate matchmaking instances

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 7450.8 | 6874.7 | 7186.8 | 7542.5 | 7435.0 |
| DLP UB | | 8643.2 | 8241.6 | 8390.8 | 8924.2 | 8715.7 |
| | lb(se) | 6822.1(27.1) | 6480.8(25.9) | 6702.4(29.7) | 6855.5(25.6) | 6869.1(31.8) |
| ALP Dual Policy | gap | 8.4 | 5.7 | 6.7 | 9.1 | 7.6 |
| ALP Primal Policy with resolve | lb(se) | 6811.5(27.1) | 6502.8(25.8) | 6688.2(30.0) | 6873.1(25.4) | 6800.1(34.1) |
| | gap | 8.6 | 5.4 | 6.9 | 8.9 | 8.5 |
| ALP Primal Policy no resolve | lb(se) | 5846.9(85.6) | 6386.6(26.8) | 6455.7(33.6) | 4957.8(105.9) | 6432.2(43.3) |
| | gap | 21.5 | 7.1 | 10.1 | 34.2 | 13.5 |
| | lb(se) | 6796.2(26.7) | 6428.3(25.8) | 6689.5(29.3) | 6840.4(25.5) | 6840.9(31.3) |
| DLP Policy | gap | 21.4 | 22.0 | 20.3 | 23.3 | 21.5 |
| | lb(se) | 6661.8(26.8) | 6335.4(25.5) | 6552.4(29.4) | 6678.2(25.4) | 6743.9(31.2) |
| LLA Policy | gap | 10.6 | 7.8 | 8.8 | 11.5 | 9.3 |

for which the column generation ALP bound is strictly tighter than the ALP reformulation bound. We observe that omitting the convex hull constraints in our relaxed ALP has a negligible impact: the bounds obtained by the relaxed ALP reformulation are nearly identical to those obtained by the ALP using column generation based ALP, and are in fact identical for the majority of instances. As for the other applications we considered, the ALP reformulation is orders of magnitudes faster. Given these results, we do not explore the possibility of strengthening the relaxed ALP by adding valid inequalities.

Table A.8: Upper and lower bounds for larger matchmaking instances

| Instance Number | | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| ALP UB | | 9577.4 | 10078.0 | 9996.9 | 10215.8 | 10033.3 |
| DLP UB | | 10206.3 | 10647.9 | 10630.1 | 10904.9 | 10567.4 |
| ALP Dual Policy | lb(se) | 9471.7(10.4) | 9999.8(10.7) | 9888.6(11.3) | 10053.5(11.5) | 9881.5(11.4) |
| | gap | 1.10 | 0.77 | 1.08 | 1.58 | 1.51 |
| ALP Primal Policy with resolve | lb(se) | 9482.7(11.4) | 10011.7(11.5) | 9887.6(11.4) | 10064.6(11.0) | 9874.5(10.8) |
| | gap | 0.98 | 0.65 | 1.09 | 1.48 | 1.58 |
| ALP Primal Policy no resolve | lb(se) | 9444.0(12.1) | 9968.6(10.8) | 9845.0(10.8) | 9948.1(12.6) | 9835.6(10.3) |
| | gap | 1.39 | 1.08 | 1.51 | 2.62 | 1.97 |
| DLP Policy | lb(se) | 9452.6(10.6) | 9974.2(11.4) | 9862.8(10.5) | 10037.6(11.4) | 9864.4(11.6) |
| | gap | 7.38 | 6.32 | 7.21 | 7.95 | 6.65 |
| LLA Policy | lb(se) | 9459.6(10.6) | 9980.2(10.8) | 9869.8(10.3) | 10032.5(10.7) | 9876.2(10.4) |
| | gap | 1.22 | 0.97 | 1.27 | 1.79 | 1.56 |

Table A.9: Patient-donor pair attributes for kidney exchange instances (Li et al. 2019)

| Attribute | Distribution |
|---|---|
| Patient Gender | M: 0.59, F: 0.41 |
| Donor is Spouse | Yes: 0.49, No: 0.51 |
| Patient Blood Type | O: 0.6, A: 0.3, B: 0.07, AB: 0.03 |
| Donor Blood Type | O: 0.46 A: 0.39 B: 0.12 AB: 0.03 |
| Patient PRA Type | High: 0.75, Low: 0.25 |

Table A.10: ALP reformulation vs. column generation for instances with $L \geq 3$

| $(V, L)$ | $(15, 3)$ | $(20, 3)$ | $(10, 4)$ |
|---|---|---|---|
| ALP time (seconds) | 0.09 | 0.27 | 0.14 |
| CG time (seconds) | 22.7 | 92.4 | 17.4 |
| ALP UB | 401.74 | 500.08 | 297.63 |
| CG UB | 401.73 | 500.07 | 297.62 |
| Avg bound difference (%) | 0.0018 | 0.0015 | 0.0013 |
| # of instances with ALP UB > CG UB | 24 | 40 | 17 |

# Appendix B

# Appendix to Chapter 3

## B.1. Additional Numerical Results

Results for all instance specifications are presented in Table B.1.

## B.2. Extension to Nested Logit

As mentioned earlier, the MNL class of choice models are often criticized for its IIA property. To alleviate the IIA issue, in certain applications, it is helpful to extend the MNL model. The nested logit (NL) model is among the most widely adopted extensions, which partitions the set of items into subsets, or "nests"; we investigate an assortment optimization problem under NL in Chapter 4. In this section, we introduce the NL model and show how it can be estimated.

For notational brevity, assume items are grouped into two nests, $F$ and $J$. Let $\mathcal{N}^F$ be the set of all $F$ items and $\mathcal{N}^J$ be the set of all $J$ items. Let $K$ be the display capacity for both $F$ and $J$ nests, that is, $K$ number of $F$ items and $K$ number of $J$ items are displayed at the same time. An arriving customer decides to either make a purchase in one of the two nests, or leave without purchase. For a visualization of the customer's choice process, see Figure 4.3. If the customer decides to make a purchase in one of the nests, he or she has to choose one of the offered items.

After a cold-start period, transactional data can be collected for choice model estimation; a sample of the transactional data is shown in Table B.2. To account for customer preference heterogeneity, one could conduct market segmentation based on transactional data using the k-means approach. We describe nested logit model estimation for each segment after customer segmentation is complete. For a generic customer segment, let $v_i^F$ be the preference weight of item $i \in \mathcal{N}^F$, and $V_j^J$ is the preference weight of item $j \in \mathcal{N}^J$. Let $\mathcal{S}^F$ and $\mathcal{S}^J$ be assortments of items shown to an arriving customer. Under NL, the probability that the customer chooses to make a purchase in $F$ is given by

$$\frac{(\sum_{i\in\mathcal{S}^F} v_i^F)^{\gamma^F}}{1+(\sum_{i\in\mathcal{S}^F} v_i^F)^{\gamma^F}+(\sum_{j\in\mathcal{S}^J} v_j^J)^{\gamma^J}},$$

the probability that the customer chooses from the $J$ assortment is

$$\frac{(\sum_{j\in\mathcal{S}^J} v_j^J)^{\gamma^J}}{1+(\sum_{i\in\mathcal{S}^F} v_i^F)^{\gamma^F}+(\sum_{j\in\mathcal{S}^J} v_j^J)^{\gamma^J}},$$

and leaves without purcahse with probability

$$\frac{1}{1+(\sum_{i\in\mathcal{S}^F} v_i^F)^{\gamma^F}+(\sum_{j\in\mathcal{S}^J} v_j^J)^{\gamma^J}}.$$

Parameters $\gamma^F, \gamma^J$ are a measure of the degree of independence of unobserved utilities among the alternatives within nests, higher values of $\gamma^F, \gamma^J$ indicate more dissimilarity for the alternatives within nests. In order to keep the model within the umbrella of RUMs, we require that dissimilarities are at most 1; see Davis et al. (2014) for a detailed discussion. In NL estimation, $\gamma^F$ and $\gamma^J$ are parameters that need to be estimated.

Under NL, if a customer decides to choose from the $F$ assortment, the the conditional probability that he/she chooses item $i \in \mathcal{S}^F$ is $v_i^F/(\sum_{i\in\mathcal{S}^F} v_i^F)^{\gamma^F}$. Similarly, if a customer decides to choose from the $J$ assortment, the the conditional probability that he/she chooses item $j \in \mathcal{S}^J$ is $v_j^J/(\sum_{j\in\mathcal{S}^J} v_j^J)^{\gamma^J}$.

Given the above, parameters of the NL model, namely utilities $v_i^F$ for $i \in \mathcal{N}^F$ and $v_j^J$ for $j \in \mathcal{N}^J$, as wells dissimilarities $\gamma^F$ and $\gamma^J$, can be estimated by optimizing the log-likelihood function.

To construct the log-likelihood function $\mathcal{LL}(\mathbf{v}^F, \mathbf{v}^J, \gamma^F, \gamma^J)$, for each row $t$ of the transactional data, let $\mathcal{S}_t^F, \mathcal{S}_t^J$ and $c_t$ be the $F$ assortment, $J$ assortment and customer choice, respectively. If $c_t = 0$, that is, the customers chooses the no-purchase option, we add the following term

$$-\log\left(1+(\sum_{i\in\mathcal{S}_t^F} v_i^F)^{\gamma^F}+(\sum_{j\in\mathcal{S}_t^J} v_j^J)^{\gamma^J}\right)$$

to $\mathcal{LL}(\mathbf{v}^F, \mathbf{v}^J, \gamma^F, \gamma^J)$. Alternatively, if $c_t \in \mathcal{S}_t^F$, we add

$$(\gamma^F - 1)\log\left(\sum_{i\in\mathcal{S}_t^F} v_i^F\right) + \log(v_{c_t}^F) - \log\left(1 + (\sum_{i\in\mathcal{S}_t^F} v_i^F)^{\gamma^F} + (\sum_{j\in\mathcal{S}_t^J} v_j^J)^{\gamma^J}\right)$$

to $\mathcal{LL}(\mathbf{v}^F, \mathbf{v}^J, \gamma^F, \gamma^J)$. If $c_t \in \mathcal{S}_t^J$, add

$$(\gamma^J - 1)\log\left(\sum_{j\in\mathcal{S}_t^J} v_j^J\right) + \log(v_{c_t}^J) - \log\left(1 + (\sum_{i\in\mathcal{S}_t^F} v_i^F)^{\gamma^F} + (\sum_{j\in\mathcal{S}_t^J} v_j^J)^{\gamma^J}\right)$$

to $\mathcal{LL}(\mathbf{v}^F, \mathbf{v}^J, \gamma^F, \gamma^J)$. Once the log-likelihood function is constructed, standard nonlinear optimization packages can be used for parameter estimation. Note that incorporating terms that account for utilities based on customer inventory is straightforward, similar to approaches described in Section 3.3.2 and 3.3.3.

As noted in previous work, $\mathcal{LL}(\mathbf{v}^F, \mathbf{v}^J, \gamma^F, \gamma^J)$ does not have much structure and is challenging to optimize. To overcome this, various approaches have been proposed, we refer to the E-companion of Berbeglia et al. (2021) for a detailed discussion. In the numerical experiments of Berbeglia et al. (2021), estimation of the NL model is about one order of magnitude slower than their MNL counterparts. Due to time constraints, in our numerical study, we do not implement and test the performance of the inventory-aware NL model.

Table B.1: Detailed results

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.384 | 384.0 |
| MNL-Seg | 0.025 (93.3) | 662.5 (72.5) |
| IA-MNL | 0.031 (91.8) | 661.4 (72.2) |
| CF | — | 360.7 (-6.06) |

(a) $(k, p, K, T) = (10, 0.2, 3, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.390 | 397.5 |
| MNL-Seg | 0.317 (18.8) | 482.5 (21.3) |
| IA-MNL | 0.318 (18.4) | 515.3 (29.6) |
| CF | — | 310.2 (-21.9) |

(b) $(k, p, K, T) = (100, 0.2, 3, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.218 | 115.9 |
| MNL-Seg | 0.093 (57.1) | 164.2 (41.5) |
| IA-MNL | 0.079 (63.6) | 166.2 (43.3) |
| CF | — | 90.98 (-21.5) |

(c) $(k, p, K, T) = (10, 0.8, 3, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.219 | 100.1 |
| MNL-Seg | 0.167 (23.9) | 115.3 (15.2) |
| IA-MNL | 0.173 (20.9) | 117.6 (17.5) |
| CF | — | 65.91 (-34.1) |

(d) $(k, p, K, T) = (100, 0.8, 3, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.332 | 403.3 |
| MNL-Seg | 0.020 (93.8) | 612.7 (51.8) |
| IA-MNL | 0.033 (89.8) | 611.3 (51.5) |
| CF | — | 349.1 (-13.4) |

(e) $(k, p, K, T) = (10, 0.2, 5, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.336 | 438.3 |
| MNL-Seg | 0.282 (15.9) | 475.5 (8.48) |
| IA-MNL | 0.275 (18.0) | 497.5 (13.4) |
| CF | — | 319.4 (-27.1) |

(f) $(k, p, K, T) = (100, 0.2, 5, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.188 | 110.4 |
| MNL-Seg | 0.053 (71.4) | 153.6 (39.1) |
| IA-MNL | 0.045 (75.6) | 156.1 (41.3) |
| CF | — | 89.57 (-18.9) |

(g) $(k, p, K, T) = (10, 0.8, 5, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.189 | 111.6 |
| MNL-Seg | 0.151 (20.4) | 114.4 (2.45) |
| IA-MNL | 0.154 (18.7) | 114.9 (2.94) |
| CF | — | 70.73 (-36.6) |

(h) $(k, p, K, T) = (100, 0.8, 5, 20)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.312 | 310.0 |
| MNL-Seg | 0.007 (97.4) | 520.9 (68.0) |
| IA-MNL | 0.013 (95.7) | 520.7 (67.9) |
| CF | — | 328.2 (5.856) |

(i) $(k, p, K, T) = (10, 0.2, 3, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.316 | 271.0 |
| MNL-Seg | 0.248 (21.4) | 351.5 (29.7) |
| IA-MNL | 0.216 (31.3) | 414.8 (53.0) |
| CF | — | 265.4 (-2.07) |

(j) $(k, p, K, T) = (100, 0.2, 3, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.154 | 76.18 |
| MNL-Seg | 0.017 (88.6) | 132.8 (74.3) |
| IA-MNL | 0.015 (89.7) | 132.7 (74.1) |
| CF | — | 82.15 (7.828) |

(k) $(k, p, K, T) = (10, 0.8, 3, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.155 | 66.48 |
| MNL-Seg | 0.130 (16.0) | 84.39 (26.9) |
| IA-MNL | 0.120 (22.3) | 94.20 (41.7) |
| CF | — | 54.70 (-17.7) |

(l) $(k, p, K, T) = (100, 0.8, 3, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.255 | 251.1 |
| MNL-Seg | 0.003 (98.4) | 365.7 (45.6) |
| IA-MNL | 0.014 (94.3) | 365.9 (45.7) |
| CF | — | 249.3 (-0.70) |

(m) $(k, p, K, T) = (10, 0.2, 5, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.260 | 253.6 |
| MNL-Seg | 0.214 (17.6) | 268.9 (6.00) |
| IA-MNL | 0.178 (31.4) | 309.5 (22.0) |
| CF | — | 223.6 (-11.8) |

(n) $(k, p, K, T) = (100, 0.2, 5, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.120 | 64.11 |
| MNL-Seg | 0.006 (94.3) | 93.67 (46.1) |
| IA-MNL | 0.011 (90.4) | 93.58 (45.9) |
| CF | — | 64.83 (1.123) |

(o) $(k, p, K, T) = (10, 0.8, 5, 40)$

| | RMSE | Revenue |
|---|---|---|
| MNL | 0.123 | 63.46 |
| MNL-Seg | 0.109 (11.3) | 67.51 (6.36) |
| IA-MNL | 0.096 (22.3) | 73.58 (15.9) |
| CF | — | 47.80 (-24.6) |

(p) $(k, p, K, T) = (100, 0.8, 5, 40)$

Table B.2: Sample in-sample transactional data for nested logit

| customer id | inventory | F assortment | J assortment | choice |
|---|---|---|---|---|
| 1 | [ ] | [F1, F2, F3] | [J1, J2, J3] | F2 |
| 1 | [F2] | [F1, F3, F5] | [J1, J3, J4] | J3 |
| 1 | [F2, J3] | [F1, F4, F5] | [J1, J2, J5] | 0 |
| 1 | [F2, J3] | [F1, F3, F5] | [J2, J5, J7] | J7 |
| 1 | [F2, J3, J7] | [F3, F4, F5] | [J4, J5, J6] | 0 |

# Appendix C

# Appendix to Chapter 4

## C.1. Additional Examples with Featured vs JFY

**JD.com.** JD.com is a Chinese e-commerce company, and one of the largest B2C retailers in the world. In Figure C.1, JD.com offers both personalized sections and non-personalized recommendations.

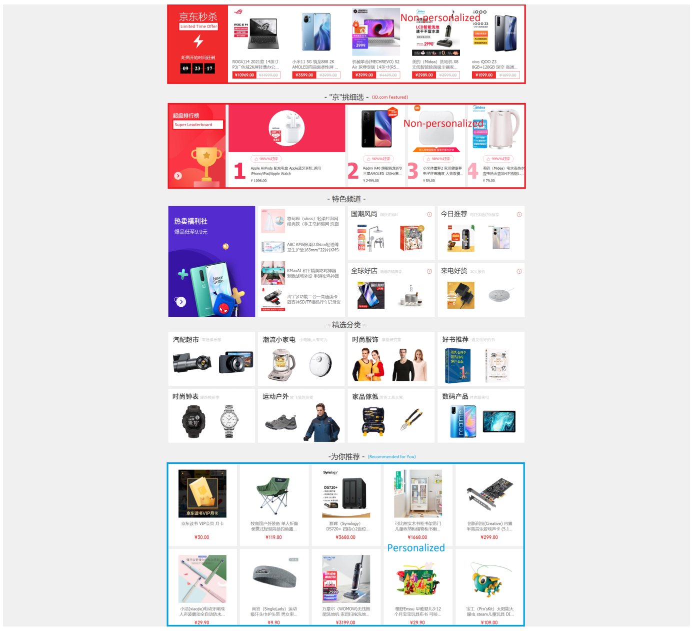


Figure C.1: JD.com recommendations

**Amazon.com.** Amazon.com is the world's largest online marketplace. As we can see in Figure C.2, Amazon offer its customers non-personalized and personalized recommendations.

**Netflix.** Netflix is the leading content streaming platform with over 200 million subscribers as of July 2021. As we can see in Figure C.3, Netflix uses a non-personalized section for promoting popular shows, and personalized sections to cater to customers' individual tastes.

Figure C.2: Amazon.com recommendations

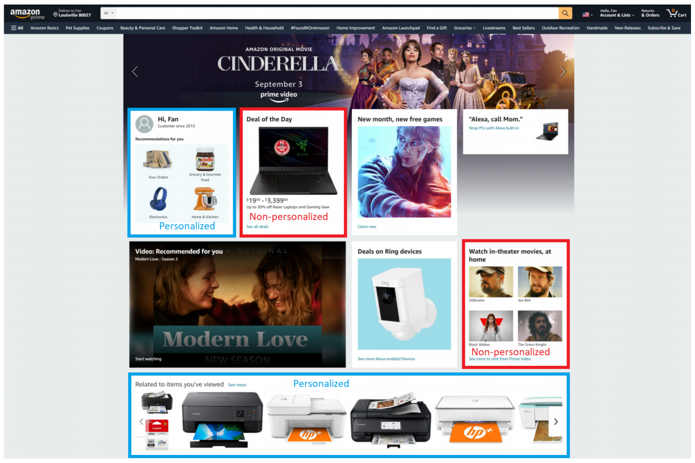


Figure C.3: Netflix recommendations

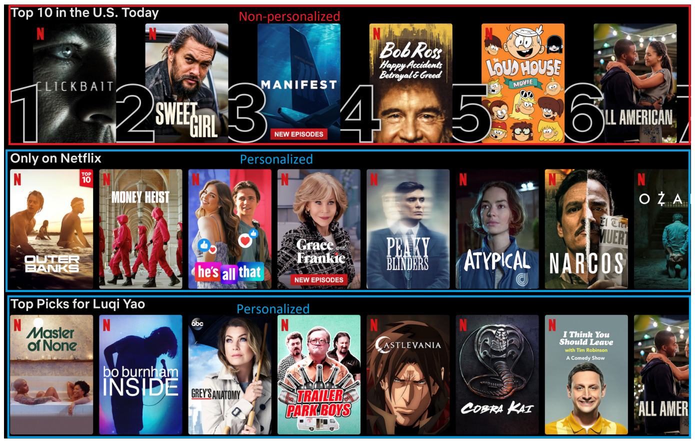
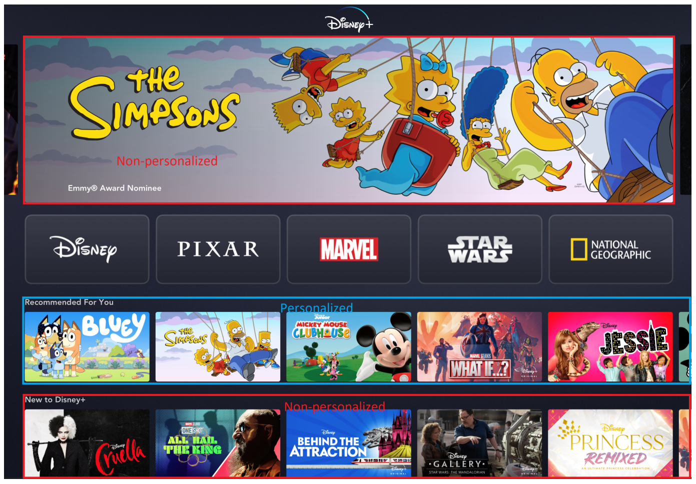


Figure C.4: Disney+ recommendations



**Disney Plus.** Disney+ is also a subscription based on-demand content streaming platform operated by the Walt Disney Company. As of July 2021, it has 116 million paid subscribers. As shown in Figure C.4, Disney+ also utilizes both non-personalized and personalized sections.

## C.2. Technical Proofs

### C.2.1 Proof of Theorem 4.1

For the sake of completeness, we first introduce relevant results from Rusmevichientong et al. (2010) and Gallego and Topaloglu (2014) using notations in this work without proof. The following Lemma shows how we can stitch together candidate Featured and JFY assortments.

**Lemma C.1.** *(Theorem 2, Gallego and Topaloglu 2014) For any $s \in \mathcal{S}$, assume we have a collection* *of candidate Featured assortments $\mathcal{C}^F$ and a collection of candidate JFY assortments $\mathcal{C}^J_s$.*

$$
\begin{aligned}
(\textbf{\textit{1-SEG}}) \quad \min \quad & z_s \\
s.t. \quad & z_s \geq y_s^F + y_s^J, \\
& y_s^F \geq \left(\sum_i v_{si}^F a_i^F\right)^{\gamma_s^F} \left(\frac{\sum_i v_{si}^F a_i^F r_i^F}{\sum_i v_{si}^F a_i^F} - z_s\right), \quad \forall \mathbf{a}^F \in \mathcal{C}^F, \\
& y_s^J \geq \left(\sum_j v_{sj}^J a_{sj}^J\right)^{\gamma_s^J} \left(\frac{\sum_j v_{sj}^J a_{sj}^J r_j^J}{\sum_j v_{sj}^J a_{sj}^J} - z_s\right), \quad \forall \mathbf{a}_s^J \in \mathcal{C}_s^J.
\end{aligned}
$$

*Let $z_s^*$ be the optimal objective value of (**1-SEG**), and let $(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J)$ be such that*

$$
\begin{aligned}
\hat{\mathbf{a}}^F &= \arg\max_{\mathbf{a}^F \in \mathcal{C}^F} \left(\sum_i v_{si}^F a_i^F\right)^{\gamma_s^F} \left(\frac{\sum_i v_{si}^F a_i^F r_i^F}{\sum_i v_{si}^F a_i^F} - z_s^*\right), \\
\hat{\mathbf{a}}^J &= \arg\max_{\mathbf{a}_s^J \in \mathcal{C}_s^J} \left(\sum_j v_{sj}^J a_{sj}^J\right)^{\gamma_s^J} \left(\frac{\sum_j v_{sj}^J a_{sj}^J r_j^J}{\sum_j v_{sj}^J a_{sj}^J} - z_s^*\right),
\end{aligned}
$$

*then $(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J)$ provides the largest expected revenue for segment $s$ customers among all candidate assortments.*

Consider any segment $s \in \mathcal{S}$ independently, suppose $(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}_s^J)$ is an optimal assortment that maximizes the total expected revenue, if we construct $\mathcal{C}^F$ and $\mathcal{C}_s^J$ such that $\hat{\mathbf{a}}^F \in \mathcal{C}^F$ and $\hat{\mathbf{a}}_s^J \in \mathcal{C}_s^J$, we will be able to obtain the optimal solution by solving a simple linear program. The next lemma characterizes good candidate assortments.

**Lemma C.2.** *(Theorem 4, Gallego and Topaloglu 2014) Consider any segment $s \in \mathcal{S}$, assume that $\mathcal{C}^F$ includes an optimal solution to optimization problem (C.1) for any $u \in \mathbb{R}_+$, and $\mathcal{C}_s^J$ includes an optimal solution to (C.2) for any $u \in \mathbb{R}_+$*

$$
\max_{\mathbf{a}^F : 1 \leq \sum_i a_i^F \leq K} \quad \left(\sum_i v_{si}^F a_i^F\right) \left(\frac{\sum_i v_{si}^F a_i^F r_i^F}{\sum_i v_{si}^F a_i^F} - u\right), \tag{C.1}
$$

$$
\max_{\mathbf{a}_s^J : 1 \leq \sum_j a_{sj}^J \leq K} \quad \left(\sum_j v_{sj}^J a_{sj}^J\right) \left(\frac{\sum_j v_{sj}^J a_{sj}^J r_j^J}{\sum_j v_{sj}^J a_{sj}^J} - u\right), \tag{C.2}
$$

*then there exists an assortment* $(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J_s)$ *with* $\hat{\mathbf{a}}^F \in \mathcal{C}^F, \hat{\mathbf{a}}^J_s \in \mathcal{C}^J_s$ *which maximizes total expected revenue from segment* $s$*.*

With Lemma C.1 and C.2, if we can construct polynomially sized candidate assortments efficiently, then we can solve the subproblems for all segments efficiently. We introduce an algorithm in Rusmevichientong et al. (2010), called the `StaticMNL` algorithm, that achieves exactly that.

**Lemma C.3.** *(Theorem 2.2, Rusmevichientong et al. 2010) The* `StaticMNL` *algorithm produces* $\mathcal{O}(K|\mathcal{N}^F|)$ *candidate assortments which include an optimal solution for (C.1) for all* $u \in \mathbb{R}_+$ *in* $\mathcal{O}(|\mathcal{N}^F|^2)$ *time, and* $\mathcal{O}(K|\mathcal{N}^J|)$ *candidate assortments which include an optimal solution for (C.2) for all* $u \in \mathbb{R}_+$ *in* $\mathcal{O}(|\mathcal{N}^J|^2)$ *time.*

We now state the `FixFeaturedSolveJFY` algorithm.

**Algorithm 7** `FixFeaturedSolveJFY`

```
Require: Instance data, fixed Featured assortment $\hat{\mathbf{a}}^F$
Ensure: JFY assortments $\hat{\mathbf{a}}^J = (\hat{\mathbf{a}}^J_1, ..., \hat{\mathbf{a}}^J_{|\mathcal{S}|})$
  for $s \in \mathcal{S}$ do
    Candidate segment $s$ JFY assortments $\mathcal{C}^J_s \leftarrow$ StaticMNL($\mathbf{v}^J_s, \mathbf{r}^J, K$)
    Formulate and solve (1-SEG) with $\mathcal{C}^F = \{\hat{\mathbf{a}}^F\}$ and $\mathcal{C}^J_s$, let $\tilde{\mathbf{a}}^J_s$ denote its optimal solution
    $\hat{\mathbf{a}}^J_s \leftarrow \tilde{\mathbf{a}}^J_s$
  end for
```

Combining results from Lemma (C.1) to (C.3), it follows that the `FixFeaturedSolveJFY` algorithm finds optimal segment-wise JFY assortments in polynomial time. ■

### C.2.2 Proof of Lemma 4.3

For part 1, for all segments $s \in \mathcal{S}$, define function

$$f_s(\alpha_s, \beta_s, \mathbf{a}^J_s) := \frac{\alpha_s^{\gamma^F_s - 1}\beta_s + \left(\sum_j a^J_{sj} v^J_{sj}\right)^{\gamma^J_s - 1} \sum_j a^J_{sj} r^J_j v^J_{sj}}{1 + \alpha_s^{\gamma^F_s} + \left(\sum_j a^J_{sj} v^J_{sj}\right)^{\gamma^J_s}}$$

for all $\mathbf{a}^J_s$. It follows that

$$z_s(\alpha_s, \beta_s) = \max_{\mathbf{a}^J_s} f_s(\alpha_s, \beta_s, \mathbf{a}^J_s).$$

Clearly $f_s(\alpha_s, \beta_s, \mathbf{a}^J_s)$ is continuous on $\Delta_s$ for all $\mathbf{a}^J_s$, and the pointwise maximum of continuous functions is also continuous.

For part 2, note that $\frac{\partial f_s}{\partial \alpha_s}$ equals

$$\frac{(\gamma_s^F - 1)\rho\alpha_s^{\gamma_s^F - 2}\left(1 + \alpha_s^F\gamma_s^F + \left(\sum_j a_{sj}^J v_{sj}^J\right)^{\gamma_s^J}\right) - \left(\alpha_s^{\gamma_s^F - 1}\beta_s + \left(\sum_j a_{sj}^J v_{sj}^J\right)^{\gamma_s^J - 1}\left(\sum_j a_{sj}^J r_j^J v_{sj}^J\right)(\gamma_s^F\alpha_s^{\gamma_s^F - 1})\right)}{\left(1 + \alpha_s^{\gamma_s^F} + \left(\sum_j a_{sj}^J v_{sj}^J\right)^{\gamma_s^J}\right)^2},$$

which given $0 < \gamma_s^F \leq 1$, implies that $\frac{\partial f_s}{\partial \alpha_s} < 0$, thus $f_s(\alpha_s, \beta_s, \mathbf{a}_s^J)$ is decreasing for all $\mathbf{a}_s^J$. Since the point-wise maximum of decreasing functions is also decreasing, we have that $z_s(\alpha_s, \beta_s)$ is decreasing in $\alpha_s$ on $\Delta_s$.

For part 3, note that $\frac{\partial f_s}{\partial \beta_s} = \alpha_s^{\gamma_s^F - 1} > 0$, which indicates that $f_s(\alpha_s, \beta_s, \mathbf{a}_s^J)$ is increasing for all $\mathbf{a}_s^J$. Because the point-wise maximum of increasing functions is also increasing, $z_s(\alpha_s, \beta_s)$ is increasing in $\beta_s$.

For part 4, let $\mathbf{a}_s^{J*}$ be an optimal solution to $\max_{\mathbf{a}_s^J} f_s(\alpha_s, \beta_s, \mathbf{a}_s^J)$. We have the following chain of inequalities

$$\begin{aligned}
c_1/c_2 \cdot z_s(c_1\alpha_s, c_2\beta_s) &\geq c_1/c_2 \cdot f_s\left(c_1\alpha_s, c_2\beta_s, \mathbf{a}_s^{J*}\right) \\
&= \frac{c_1^{\gamma_s^F}\alpha_s^{\gamma_s^F - 1}\beta_s + c_1/c_2 \cdot (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J - 1}\sum_j v_{sj}^J a_j^{J*} r_j^J}{1 + c_1^{\gamma_s^F}\alpha_s^{\gamma_s^F} + (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J}} \\
&\geq \frac{c_1^{\gamma_s^F}\alpha_s^{\gamma_s^F - 1}\beta_s + c_1/c_2 \cdot (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J - 1}\sum_j v_{sj}^J a_j^{J*} r_j^J}{c_1^{\gamma_s^F} + c_1^{\gamma_s^F}\alpha_s^{\gamma_s^F} + c_1^{\gamma_s^F}(\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J}} \\
&= \frac{\alpha_s^{\gamma_s^F - 1}\beta_s + c_1^{1-\gamma_s^F}/c_2 \cdot (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J - 1}\sum_j v_{sj}^J a_j^{J*} r_j^J}{1 + \alpha_s^{\gamma_s^F} + (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J}} \\
&\geq \frac{\alpha_s^{\gamma_s^F - 1}\beta_s + (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J - 1}\sum_j v_{sj}^J a_j^{J*} r_j^J}{1 + \alpha_s^{\gamma_s^F} + (\sum_j v_{sj}^J a_j^{J*})^{\gamma_s^J}} \\
&= f_s(\alpha_s, \beta_s, \mathbf{a}_s^{J*}) = z_s(\alpha_s, \beta_s),
\end{aligned}$$

where the first inequality follows from the fact that $\mathbf{a}_s^{J*}$ is a feasible but not necessarily optimal solution for $\max_{\mathbf{a}_s^J} f_s(c_1\alpha_s, c_2\beta_s, \mathbf{a}_s^J)$, the second inequality holds because $c_1^{\gamma_s^F} \geq 1$, and the last inequality follows from the assumption that $0 < \gamma_s^F \leq 1$ which means $c_1^{1-\gamma_s^F} > 1$.

To see that part 5 is true, first note that $z_s(\alpha_s', \beta_s') \leq z_s(\alpha_s, \beta_s)$ holds because of monotonicity.

Next, we apply part 4 by setting $c_1 = 1+\epsilon$ and $c_2 = 1/(1+\epsilon)$, which leads to

$$(1+\epsilon)^2 z_s(\alpha'_s, \beta'_s) = (1+\epsilon)^2 z_s\left((1+\epsilon)\alpha'_s, \frac{1}{1+\epsilon}\beta'_s\right) \geq z_s(\alpha_s, \beta_s)$$

reaching the desired result. ■

### C.2.3 Proof of Lemma 4.4

For (a), note that for any $\mathbf{a}^F$ that satisfy (4.3) and (4.5), it holds that

$$\begin{aligned}\sum_i \left\lceil \frac{K v^F_{si}}{\epsilon \check{\alpha}_s} \right\rceil a^F_i &\leq \sum_i \left(\frac{K v^F_{si}}{\epsilon \check{\alpha}_s} + 1\right) a^F_i \\ &= \frac{K}{\epsilon \check{\alpha}_s} \sum_i v^F_{si} a^F_i + \sum_i a^F_i \\ &\leq \frac{K}{\epsilon \check{\alpha}_s} \check{\alpha}_s + K \leq \left\lceil \frac{K}{\epsilon} \right\rceil + K;\end{aligned}$$

on the other hand, for any $\mathbf{a}^F$ that satisfy (4.4) and (4.5), we have

$$\begin{aligned}\sum_i \left\lfloor \frac{K v^F_{si} r^F_i}{\epsilon \check{\beta}_s} \right\rfloor a^F_i &\geq \sum_i \left(\frac{K v^F_{si} r^F_i}{\epsilon \check{\beta}_s} - 1\right) a^F_i \\ &= \frac{K}{\epsilon \check{\beta}_s} \sum_i v^F_{si} a^F_i r^F_i - \sum_i a^F_i \\ &\geq \frac{K}{\epsilon \check{\beta}_s} \check{\beta}_s - K \geq \left\lfloor \frac{K}{\epsilon} \right\rfloor - K.\end{aligned}$$

To prove (b), note that for any $\mathbf{a}^F$ that satisfy (4.6), the following holds

$$\frac{K}{\epsilon \check{\alpha}_s} \sum_i v^F_{si} a^F_i \leq \sum_i \left\lceil \frac{K v^F_{si}}{\epsilon \check{\alpha}_s} \right\rceil a^F_i \leq \left\lceil \frac{K}{\epsilon} \right\rceil + K \leq \frac{K}{\epsilon} + 1 + K,$$

which suggests

$$\sum_i v^F_{si} a^F_i \leq \left(\frac{K}{\epsilon} + 1 + K\right) \cdot \frac{\epsilon \check{\alpha}_s}{K} = \left(1 + \frac{\epsilon}{K} + \epsilon\right) \check{\alpha}_s \leq (1+2\epsilon)\check{\alpha}_s.$$

Similarly for (c), for any $\mathbf{a}^F$ that satisfy (4.7), we have

$$\frac{K}{\epsilon \check{\beta}_s} \sum_i v^F_{si} a^F_i r^F_i \geq \sum_i \left\lfloor \frac{K v^F_{si} r^F_i}{\epsilon \check{\beta}_s} \right\rfloor a^F_i \geq \left\lfloor \frac{K}{\epsilon} \right\rfloor - K \geq \frac{K}{\epsilon} - 1 - K,$$

therefore

$$\sum_i v_{si}^F a_i^F r_i^F \geq (\frac{K}{\epsilon} - 1 - K) \cdot \frac{\epsilon \check{\beta}_s}{K} = (1 - \frac{\epsilon}{K} - \epsilon)\check{\beta}_s \geq (1 - 2\epsilon)\check{\beta}_s.$$

$\blacksquare$

### C.2.4 Proof of Theorem 4.5

Let $\mathbf{a}^{F*}$ be an optimal solution to (4.1), and denote $\alpha_s^* := \sum_i v_{si}^F a_i^{F*}$ and $\beta_s^* := \sum_i v_{si}^F r_i^F a_i^{F*}$ for all $s$. In addition, let $(g_s^*, h_s^*), \forall s$ represent the parameter grid that corresponds to $\mathbf{a}^{F*}$, in other words

$$\alpha_{s,g_s^*} \leq \alpha_s^* \leq \alpha_{s,g_s^*+1}, \quad \beta_{s,h_s^*} \leq \beta_s^* \leq \beta_{s,h_s^*+1}, \quad \forall s.$$

Note that because $\mathbf{a}^{F*}$ is feasible for parameter $(\alpha_{1,g_1^*+1}, ..., \alpha_{|\mathcal{S}|,g_{|\mathcal{S}|}^*+1}, \beta_{1,h_1^*}, ..., \beta_{|\mathcal{S}|,h_{|\mathcal{S}|}^*})$, by (a), it is feasible for the DP subproblem, thus the DP subroutine returns an approximately feasible Featured assortment, denoted $\tilde{\mathbf{a}}^F$, and let $\tilde{\mathbf{a}}^J = \texttt{FixFeaturedSolveJFY}(\tilde{\mathbf{a}}^F)$. By (b) and (c) in Lemma 4.4, we have

$$\sum_i v_{si}^F \tilde{a}_i^F \leq (1+2\epsilon)\alpha_{s,g_s^*+1} = (1+\epsilon)(1+2\epsilon)\alpha_{s,g_s^*} \leq (1+\epsilon)(1+2\epsilon)\alpha_s^*,$$

$$\sum_i v_{si}^F r_i^F \tilde{a}_i^F \geq (1-2\epsilon)\beta_{s,h_s^*} = (1-2\epsilon)/(1+\epsilon)\beta_{s,h_s^*+1} \geq (1-2\epsilon)/(1+\epsilon)\beta_s^*.$$

According to Lemma 4.3, it holds that

$$\begin{aligned} &\frac{(1+\epsilon)^2(1+2\epsilon)}{1-2\epsilon} \cdot z_s\left(\sum_i v_{si}^F \tilde{a}_i^F, \sum_i v_{si}^F r_i^F \tilde{a}_i^F\right) \\ &\geq \frac{(1+\epsilon)^2(1+2\epsilon)}{1-2\epsilon} \cdot z_s\left((1+\epsilon)(1+2\epsilon)\alpha_s^*, \frac{1-2\epsilon}{1+\epsilon}\beta_s^*\right) \\ &\geq z_s(\alpha_s^*, \beta_s^*). \end{aligned}$$

Putting everything together, we have

$$\begin{aligned}\Pi(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J) &\geq \Pi(\tilde{\mathbf{a}}^F, \tilde{\mathbf{a}}^J) \\ &= \sum_s m_s z_s \left( \sum_i v_{si}^F \tilde{a}_i^F, \sum_i v_{si}^F r_i^F \tilde{a}_i^F \right) \\ &\geq \frac{1-2\epsilon}{(1+\epsilon)^2(1+2\epsilon)} \sum_s m_s z_s(\alpha_s^*, \beta_s^*) \\ &= \frac{1-2\epsilon}{(1+\epsilon)^2(1+2\epsilon)} \cdot Z^*,\end{aligned}$$

which means Algorithm 1 returns a $1-\mathcal{O}(\epsilon)$ approximate solution. Next we show that the running time is polynomial. Note that we create $G_1 \cdot H_1 \cdots G_{|\mathcal{S}|} \cdot H_{|\mathcal{S}|}$ parameter combinations, which is in the order of $\mathcal{O}\left((\log A_{\max})^{2|\mathcal{S}|}(\frac{1}{\epsilon})^{2|\mathcal{S}|}\right)$; for each parameter combination, the DP subproblem has $\mathcal{O}\left(|\mathcal{N}^F| \cdot K^{2|\mathcal{S}|} \cdot (\frac{1}{\epsilon})^{2|\mathcal{S}|}\right)$ states. Therefore, the running time of Algorithm 1 is in the order of $\mathcal{O}\left((\log A_{\max})^{2|\mathcal{S}|} \cdot |\mathcal{N}^F| \cdot K^{2|\mathcal{S}|} \cdot (\frac{1}{\epsilon})^{4|\mathcal{S}|}\right)$. ■

### C.2.5 Proof of Theorem 4.6

The true optimal solution is in some grid. The PL approximation in that grid will differ no more than a factor of $(1+\epsilon)^2$. The optimal solution of the PL integer program produces a solution better than $\frac{1}{(1+\epsilon)^2}$ of the true optimal solution, which in turn indicates that the true objective value of the PL solution is at least $\frac{1}{(1+\epsilon)^4}$ of the true optimal. In other words $(1+\epsilon)^4$ is guaranteed for the worst case when (**PLA**) overestimates its optimal solution and underestimate the true optimal solution. ■

### C.2.6 Proof of Proposition 4.7

Denote $\pi_s(\mathbf{a}^F)$ as the optimal expected revenue for segment $s$ customers when the Featured assortment is $\mathbf{a}^F$, and let $\tilde{\mathbf{a}}_s^F$ be the Featured assortment that maximizes $\pi_s(\mathbf{a}^F)$. Without loss of generality, relabel the segments such that $m_1\pi_1(\tilde{\mathbf{a}}_1^F) \geq m_s\pi_s(\tilde{\mathbf{a}}_s^F), \forall s$. Moreover, define $\tilde{\mathbf{a}}^J =$

`FixFeaturedSolveJFY(`$\tilde{\mathbf{a}}_1^F$`)`. With the above definitions, we have the following chain of inequalities

$$\begin{aligned}
|\mathcal{S}| \cdot \Pi(\hat{\mathbf{a}}^F, \hat{\mathbf{a}}^J) &\geq |\mathcal{S}| \cdot \Pi(\tilde{\mathbf{a}}_1^F, \tilde{\mathbf{a}}^J) \\
&= |\mathcal{S}| \cdot \sum_s m_s \pi_s(\tilde{\mathbf{a}}_1^F) \\
&\geq |\mathcal{S}| \cdot m_s \pi_1(\tilde{\mathbf{a}}_1^F) \\
&\geq \sum_s m_s \pi_s(\tilde{\mathbf{a}}_s^F) \\
&\geq Z^*,
\end{aligned}$$

which gives the approximation result. ■

### C.2.7 Proof of Lemma 4.8

Note that an optimal solution $(\mathbf{a}^{F*}, \mathbf{a}_s^{J*})$ to (4.1) is feasible but not necessarily optimal for (**LagSub**). Let $(\hat{\mathbf{a}}_s^F, \hat{\mathbf{a}}_s^J)$ be an optimal solution to the (**LagSub**) problem for segment $s$. It follows that

$$\begin{aligned}
&\sum_s m_s \pi_s(\boldsymbol{\lambda}_s) \\
&= \sum_s m_s \frac{\left(\sum_i v_{si}^F \hat{a}_{si}^F\right)^{\gamma_s^F - 1} \sum_i v_{si}^F \hat{a}_{si}^F r_i^F + \left(\sum_j v_{sj}^J \hat{a}_{sj}^J\right)^{\gamma_s^J - 1} \sum_j v_{sj}^J \hat{a}_{sj}^J r_j^J}{1 + \left(\sum_i v_{si}^F \hat{a}_{si}^F\right)^{\gamma_s^F} + \left(\sum_j v_{sj}^J \hat{a}_{sj}^J\right)^{\gamma_s^J}} - \sum_s m_s \left(\sum_i \lambda_{si} \hat{a}_{si}^F\right) \\
&\geq \sum_s m_s \frac{\left(\sum_i v_{si}^F a_i^{F*}\right)^{\gamma_s^F - 1} \sum_i v_{si}^F a_i^{F*} r_i^F + \left(\sum_j v_{sj}^J a_j^{J*}\right)^{\gamma_s^J - 1} \sum_j v_{sj}^J a_j^{J*} r_j^J}{1 + \left(\sum_i v_{si}^F a_i^{F*}\right)^{\gamma_s^F} + \left(\sum_j v_{sj}^J a_j^{J*}\right)^{\gamma_s^J}} - \sum_i \hat{a}_{si}^F \left(\sum_s m_s \lambda_{si}\right) \\
&= Z^*,
\end{aligned}$$

which is the desired result. ■

### C.2.8 Proof of Lemma 4.9

We again use variable $\alpha_s$ to represent the total preference weight of Featured assortment for segment $s$. We also define $\delta_s$ as the total preference weight of the JFY assortment for segment

$s$. The first two constraints follow from the assumption that $\gamma_s^F$ and $\gamma_s^J$ are between 0 and 1, which means that $\frac{\alpha_s^{\gamma_s^F-1}}{1+\alpha_s^{\gamma_s^F}+\delta_s^{\gamma_s^J}}$ is monotonically decreasing in $\alpha_s$, and $\frac{\delta_s^{\gamma_s^J-1}}{1+\alpha_s^{\gamma_s^F}+\delta_s^{\gamma_s^J}}$ is monotonically decreasing in $\delta_s$. ■

### C.2.9 Proof of Theorem 4.10

First note that the structure of (**LagSub-Reform**) allows us to separate the formulation into two knapsack subproblems, one for Featured and one for JFY. The monotonicity of $\alpha_s^{\gamma_s^F-1}/(1+\alpha_s^{\gamma_s^F}+\delta_s^{\gamma_s^J})$ and $\delta_s^{\gamma_s^J-1}/(1+\alpha_s^{\gamma_s^F}+\delta_s^{\gamma_s^J})$ ensures that the objective function coefficients of (**SubKnap-F**) and (**SubKnap-J**) are larger, and the constraints are looser; combined with a relaxation of the integrality constraints, we have that $\phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s)$ is an upper bound on $\pi_s(\boldsymbol{\lambda}_s)$, since the $\phi_s(\cdot)$ value of the grid corresponding to the values of a true optimal solution to (**LagSub-Reform**) is ensured to be greater than or equal to $\pi_s(\boldsymbol{\lambda}_s)$. The upper bound on the overall expected revenue follows from Lemma 4.8 and 4.9. ■

### C.2.10 Proof of Lemma 4.11

First note that $\eta_s(\boldsymbol{\lambda}_s, \alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1})$ when viewed as a function of $\boldsymbol{\lambda}_s$, corresponds to the optimal objective value of a linear program as a function of objective function coefficients. As a result, following standard linear programming theory, we have that $\eta_s(\boldsymbol{\lambda}_s, \alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1})$ is convex in $\boldsymbol{\lambda}_s$. Since $\theta_s(\alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1})$ does not depend on $\boldsymbol{\lambda}_s$, and because the pointwise maximum of convex functions is also convex, $\max_{g,p}(\eta_s(\boldsymbol{\lambda}_s, \alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1}) + \theta_s(\alpha_{s,g}, \alpha_{s,g+1}, \delta_{s,p}, \delta_{s,p+1}))$ is convex.

Let $g^*$ and $p^*$ be the grid where maximum is attained, and $\hat{\mathbf{a}}_s^J$ is an optimal solution of (**SubKnap-J**). By the definition of $\phi_s(\cdot)$, we have

$$\phi_s(\hat{\boldsymbol{\lambda}}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) = \frac{\alpha_{s,g^*}^{\gamma_s^F-1}}{1+\alpha_{s,g^*}^{\gamma_s^F}+\delta_{s,p^*}^{\gamma_s^J}} \sum_i v_{si}^F r_i^F \hat{a}_{si}^F - \sum_i \hat{\lambda}_{si}^F \hat{a}_{si}^F + \frac{\delta_{s,p^*}^{\gamma_s^j-1}}{1+\alpha_{s,g^*}^{\gamma_s^F}+\delta_{s,p^*}^{\gamma_s^J}} \sum_j v_{sj}^J r_j^J \hat{a}_{sj}^J. \quad (C.3)$$

Now consider an arbitrary $\boldsymbol{\lambda}_s$, it holds that

$$\phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) \geq \frac{\alpha_{s,g^*}^{\gamma_s^F - 1}}{1 + \alpha_{s,g^*}^{\gamma_s^F} + \delta_{s,p^*}^{\gamma_s^J}} \sum_i v_{si}^F r_i^F \hat{a}_{si}^F - \sum_i \lambda_{si}^F \hat{a}_{si}^F + \frac{\delta_{s,p^*}^{\gamma_s^j - 1}}{1 + \alpha_{s,g^*}^{\gamma_s^F} + \delta_{s,p^*}^{\gamma_s^J}} \sum_j v_{sj}^J r_j^J \hat{a}_{sj}^J, \quad \text{(C.4)}$$

where the inequality holds because $g^*$ and $p^*$ may not be the grid where maximum is attained for $\boldsymbol{\lambda}_s$, and that $\hat{\mathbf{a}}_s^F$ and $\hat{\mathbf{a}}_s^J$ are feasible but not optimal for (**SubKnap-F**) and (**SubKnap-J**) that correspond to $\boldsymbol{\lambda}_s$. Subtracting (C.3) from (C.4) yields

$$\phi_s(\boldsymbol{\lambda}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) \geq \phi_s(\hat{\boldsymbol{\lambda}}_s, \boldsymbol{\alpha}_s, \boldsymbol{\delta}_s) - \sum_i \hat{a}_{si}^F \left( \lambda_{si} - \hat{\lambda}_{si} \right),$$

which implies the desired result. ■