New Approaches for Integrating Revenue and Supply Chain Management

by

Adam Nabil Elmachtoub

B.S., Cornell University (2009)

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Author . . . .
Sloan School of Management
August 15, 2014

Certified by . . . .
Retsef Levi
J. Spencer Standish (1945) Professor of Management
Professor of Operations Management
Thesis Supervisor

Accepted by . . . .
Dimitris Bertsimas
Boeing Professor of Operations Research
Co-director of Operations Research Center

New Approaches for Integrating Revenue and Supply Chain Management

by

Adam Nabil Elmachtoub

Submitted to the Sloan School of Management
on August 15, 2014, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

First, we describe a general framework called online customer selection that describes natural settings where suppliers must actively select which customer requests to serve. Unlike traditional revenue management models that have sunk costs, we assume there are supply chain costs that depend on the demand being served. Specifically, customers arrive in an online manner, each with a set of requirements and associated revenue, and are either accepted or rejected upon arrival. Rejected customers incur a lost-sales cost, while accepted customers are satisfied with minimum possible production cost. The goal of the supplier is to minimize the total cost of lost sales and production. We provide algorithms with strong performance guarantees that are based on new variants of repeated optimization as well as concepts from mechanism design.

Second, we study the use of opaque products in a retail setting. A product is said to be opaque when one or more of its attributes are hidden until the transaction is complete. Opaque products have been used in the hotel and airline industry where customers purchase rooms or airfare without a priori knowledge of the brand name. In this work, we propose the use of opaque product selling in the retail industry, where there are nonperishable goods and supply chain costs. We show that a small amount of opaque selling can achieve significant ordering and holding costs savings for the supply chain. Moreover, we describe settings when a stationary opaque selling strategy can outperform a common dynamic pricing strategy.

Third, we focus on a variant of the joint replenishment problem, which arises in the previous two parts as well as in inventory management, logistics, and maintenance scheduling. In this problem, there are multiple item types that each has a given time-dependent sequence of demands that need to satisfied. Every time an order of item types is placed, there is an associated fixed setup cost that is submodular in the subset of item types ordered. The overall goal is to minimize the total fixed ordering costs plus inventory holding costs. We provide a variety of approximation algorithms for this problem and some special cases.

Thesis Supervisor: Retsef Levi
Title: J. Spencer Standish (1945) Professor of Management
Professor of Operations Management

Acknowledgments

The first person I must acknowledge is my advisor, Retsef Levi. Retsef has been an amazing source of personal and academic support throughout the years, and has pushed me to levels I never thought I could achieve. On numerous occasions when I thought my research could go no further, Retsef has always been able to provide me with the optimism and motivation for me to continue further. His enthusiasm, passion, creativity, and endless effort are truly inspiring, and I hope to match that throughout my own life ... within a factor of two.

I would also like to thank the other members of my thesis committee, David Shmoys and Vivek Farias. David has been an incredible source of support since my time at Cornell, and has been a wonderful mentor and coauthor since. He truly believed in me from the day we met, and has given me invaluable advice and help over the years. Although I have not had the pleasure to work directly with Vivek (yet), I have always been impressed by his seemingly endless knowledge and excitement. Besides being a great teacher, he also has the ability to always provide a unique perspective on research.

There are many other faculty and staff at MIT that need to be thanked. Georgia Perakis has been a great source of support on numerous occasions and a wonderful person to be around. I have also cherished my interactions with Itai Ashlagi, Robert Freund, David Gamarnik, James Orlin, Tauhid Zaman, and Karen Zheng around Sloan over the years. The ORC Co-Directors Dimitris Bertimas and Patrick Jaillet have always been available for me, and have done a great job in guiding the ORC over the years. Finally, thank you to Laura Rose and Andrew Carvalho who have made this process as smooth as possible.

I would also like to thank the faculty at Cornell who were my original role models and sources of encouragement. Charlie van Loan and Leslie Trotter worked with me as an undergraduate, and they made research a fun and rewarding experience. Bob Bland and Mike Todd provided excellent guidance for me while I was at Cornell, and played a large role in encouraging me to pursue academia.

This thesis has also relied on the support of very generous sponsors. For my first three years, I was funded by a National Defense Science and Engineering Graduate (NDSEG) fellowship from the Air Force of Scientific Research. Later on I also received support from the Center for Biomedical Innovation at MIT, and I would specifically like to thank Gigi Hirsch for her support.

During my time at MIT, I have met some amazing people and lifelong friends. Ross Anderson, Fernanda Bravo, Andre Calmon, Florin Ciocan, Maxime Cohen, David Goldberg, Vishal Gupta, Angie King, Kris Johnson, Aileen Malloy, Allison O'hair, Anna Papush, David Relethford, Stefano Traca, and Yehua Wei have made this journey so fun and meaningful, and thankfully have learned to tolerate me over the years. Ross was a roommate of mine for three years during which we had countless laughs, too much caffeine, and an unusual amount of venison. Aileen, Anna, David R., and Stefano have also been great roommates that made living in Cambridge feel like home. Andre and I went through almost everything together, which made it all a lot more fun. David G. and I spent many late nights at the ORC, and he has inspired me with his extreme passion for research. Florin, Vishal, Angie, Kris, and Allison are just wonderful friends that I have relied on and shared many great experiences with over they years and hopefully many more. Yehua is not only a friend but a great coauthor for our work on opaque products, which is included in this thesis. Maxime and Paul have been tremendous officemates who are always willing to listen to my thoughts and go on coffee runs with me. Last, but certainly not least, Fernanda has been an invaluable friend, to an extent that I did not know was possible.

Finally, I would like to thank my parents Nabil and Daad, and my siblings Najla, Ryan and Sarah. My father, Nabil, has always been a role model and taught me the meaning of hard work and using my brain above all else. My mother, Daad, has been a constant source of support and love and has given me the common sense I need to navigate my life. My sister Najla is an amazingly talented person who is always able to outwit me and show me how to be a cooler person. My brother Ryan is the most sincere person I know, and is always reminding me that there is more to life than books (and that I need to work out more). My sister Sarah is the type of person

that can brighten anyone's day, and she is a constant source of positivity and fashion advice. My family has always been my biggest supporters, and therefore I dedicate this thesis to them.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

In this thesis, we study new models that integrate revenue management and supply chain management decisions. Revenue management has historically been motivated by the airline and hotel industries. In these contexts, there is a typically a fixed amount of given capacity (seats/rooms) that needs to be sold, and the goal is to get as much value as possible from this capacity. Since generating the capacity occurs far in advance from when sales are generated, the costs are assumed to be sunk and the focus is solely on maximizing revenue. The goal of traditional revenue management has been to design tactics that obtain as much revenue as possible from a given capacity, using techniques such as dynamic pricing, bid-price controls, protection levels, and promotions. In traditional supply chain management, the goal is to serve a set of demand as efficiently as possible. The demand is typically assumed to be provided exogenously and is either described deterministically or via a probability distribution. The supplier has to make corresponding inventory, capacity, and logistics decisions in order to meet the demand accordingly with minimum cost to the supply chain.

In combining revenue and supply chain management systems in one framework, where applicable, we would like to simultaneously optimize both the revenues and supply chain costs required to satisfy the demand . In particular, revenue management decisions affect the revenue and demand, which consequently also impacts the supply chain costs. For example, dynamic pricing has been one of the primary vehicles to bridge revenue and supply chain management together (for example, see Chen and

Simchi-Levi [22], Federgruen and Heching [32], and Petruzzi and Dada [69]).

In this thesis, we will use two other ideas inspired by revenue management practices to influence demand and thus integrate our revenue and supply chain cost decisions: *online customer selection* and *opaque selling*. In online customer selection, different customers arrive and we must immediately accept or reject their request upon arrival. In traditional revenue management, customers request an itinerary that uses up some capacity and customers are linked together since they are vying for the same capacity. In the context we consider, a customer request incurs supply chain costs if accepted, and customer are linked together because they will implicitly share the supply chain costs due to economies of scale. Specifically, customers arrive in an online manner, each with a set of requirements and associated revenue, and are either accepted or rejected upon arrival. Rejected customers incur a lost-sales cost, while accepted customers are satisfied with minimum possible production cost. The goal of the supplier is to minimize the total cost of lost sales and production. We provide algorithms with strong performance guarantees that are based on new variants of repeated optimization as well as concepts from mechanism design. This work is detailed in Chapter 2 and the papers Elmachtoub and Levi [27] and Elmachtoub and Levi [26].

In the second part of the thesis, we study the use of opaque products in an online retail setting. A product is said to be opaque if one or more attributes of the product is hidden until the transaction is complete. In traditional revenue management applications, opaque products have been used in the hotel and airline industry where customers can purchase discount rooms or airfare without a priori knowledge of the brand name. For example, Hotwire.con and Priceline.com sell discounted rooms and flights without revealing the brand name. This allows the hotels and airlines to increase revenues and capacity utilization without cannibalizing their fully priced, transparent products. In this work, we propose the use of opaque product selling in the retail industry, where there are nonperishable goods and supply chain costs. We show that opaque selling strategies can always lead to more profit, without necessarily losing revenue. In particular, we derive simple expressions for estimating the costs, demand, and revenue under an opaque selling strategy. Using computational and theoretical evidence, we also show that a small amount of opaque selling can achieve significant ordering and holding costs savings for the supply chain. We also compare opaque selling strategies to a class of dynamic pricing policies and show that opaque selling can be more profitable under some restrictions. Finally, we describe a generalization of opaque products that provides a natural tradeoff between customer choice and supply chain performance. This work is detailed in Chapter 3 and the paper Elmachtoub and Wei [28].

In the third part of the thesis, we analyze in detail a multi-item inventory problem with joint setup costs. These type of problems arise naturally in many settings such as those in the previous two parts of the thesis. Specifically, we describe a problem where there is a known demand for several item types, and each demand must be served by an order for that item type before its due date. The costs are composed of joint ordering costs and holding costs. Typically a simple structure is used to describe the joint ordering costs, and in our work we generalize the cost structure to allow any cost that is a submodular function over the item types ordered. We provide a variety of formulations and approximation algorithms for this problem and special cases of it. This work is detailed in Chapter 4 and the paper Cheung et al. [23].

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Supply Chain and Logistics Models with Online Customers Selection

2.1 Introduction

Supply chain management theory provides many streamlined optimization models where the goal is to satisfy an exogenous deterministic or stochastic sequence of customers over a specified planning horizon at minimum cost. More recent practice and research trends in supply chain management have led to broader models that consider decisions on the supply side as well as on the demand side. More specifically, the customers to which the supply chain should respond and commit to are not entirely exogenous parameters, but may be influenced by endogenous decisions such as pricing, promotions, and other strategic marketing-based factors. In particular, a supplier should strive to optimally match demand to the supply chain's production capabilities. A fundamental aspect of this issue is the choice of customers to which the supplier commits to serving, and how these customers are implicitly, or explicitly, chosen. These decisions regarding which customers to serve often depend on the customer's associated revenue as well as the marginal costs expected to be incurred if the customer is satisfied.

We study a broad class of supply chain models that capture situations in which customers arrive to the supply chain sequentially, each with specific requirements and an associated revenue. Our core model has decisions that are made in two phases. First, there is a selection phase where the decision maker has to decide in real-time which customers to accept and which ones to reject, without assuming any knowledge of future customer arrivals. We refer to this process as *online customer selection*. After the selection decisions are made, there is a production phase where the decision maker must serve all of the accepted customers with minimum production cost. Each of the rejected customers incurs a rejection cost that can be associated with lost revenue or a fee paid to a third-party supplier to serve the "rejected" customer. The goal is to minimize the total rejection cost of rejected customers plus the production cost of satisfying the accepted customers.

The current standard for making customer selection decisions involves the use of available-to-promise (ATP) strategies. When evaluating whether or not to accept an order, the ATP function will typically check to see if the order is *feasible* by checking on-hand inventory and/or manufacturing capacity. In this work, we attempt to take ATP to the next step by checking whether or not an order will be *profitable*. Given that an order is feasible, deciding whether or not to accept the order based on the cost of satisfying the order versus the associated revenue is what we seek to answer. Since production costs of different customers' orders are typically non-separable, the task of defining the cost of a customer order can be challenging and thus motivates this work.

The online customer selection models studied in this work attempt to capture real-life operational situations of a make-to-order supplier or service provider. Typically, decisions are broken up into phases (weeks, quarters, years). In each phase, the supplier receives customer orders (requests) due in some time period in future phases, and needs to immediately (or by the end of the current phase) decide which customers to serve as their requests arrive. During the same phase, the supplier is also serving the customers that were accepted in previous phases with minimum production cost. In many of these settings, it is often extremely challenging to form a reliable probability distribution of future demands. The challenge stems from various potential sources such as market volatility, lack of reliable data, and the fact that customers have very different needs for customized products (or services). In light of these issues, the assumption that the supplier has minimal knowledge about future customer arrivals could lead to more robust policies. (Alternatively, wrong assumptions on the demand distribution could be very adversarial.)

2.1.1 Contributions

The major contributions in this work are two-fold. First, we introduce a general class of models for online customer selection problems that captures several important settings. Similar to yield management, we aim to select an optimal set of customers to maximize our profitability, although in this setting we do not have sunk costs or perishable inventory. In particular, we study policies in settings where there are economies of scale to serving customers, and determining the marginal cost of a single customer is highly dependent on the other customers that are selected. Second, we propose three novel *online* algorithms for making decisions in general online customer selection models. The performance of the proposed online algorithms is evaluated using the well-known *competitive ratio* framework, which is widely used in many optimization settings. In particular, the online policies are compared to the *optimal offline solution* that can be obtained if one has upfront knowledge on the specific customer arrivals and their demands. Moreover, the assumption is that the customer arrivals can be generated by a worst-case adversary, who aims to maximize the ratio between the cost of the online and optimal offline solutions. Under realistic assumptions, our online policies obtain small competitive ratios for a broad array of supply chain models with online customer selection. That is, the cost of the online policies is guaranteed to be within a predetermined factor of the optimal offline policy for *any* sequence of customers and their demands.

Two of the online algorithms we propose, *Copycat* and *StablePair*, are based on a new application of repeated re-optimization. A very common heuristic frequently used to make decisions in practice is to re-optimize in every period based on the current state of the system and the decisions made so far. Empirically, these types of heuristics perform well in certain settings and poorly in others. In contrast to this approach, Copycat and StablePair re-optimize assuming that previously made decisions *can be changed*, and use the resulting outcome to guide the online decisions in the current period. (As we shall show, the resulting online policy is still feasible.) Specifically, these algorithms make decisions for the online customer selection problem based on solving a problem with offline customer selection defined with respect to all the customers that arrived thus far and assuming that no selection decisions have yet been made. StablePair looks at several subproblems, but is actually computationally more efficient than Copycat which looks at one large problem that can possibly be NP-hard. The resulting offline solution is then used to make a decision regarding the selection (rejection) of the customer that just arrived. Although these re-optimization heuristics ignore previously made decisions, the accept/reject decisions that have been made cannot be reversed. Both algorithms have small constant competitive ratios and perform well in computational experiments for a variety of online customer selection problems.

The third online algorithm, *FairShare*, is based on repeatedly simulating what is known as a cost sharing mechanism. In a cost sharing mechanism, N "players" each submit a bid representing how much they would like to be served. Then, the mechanism decides which set of players to serve and how much each player should pay according to a *cost sharing method.* For the algorithm, all previously observed customers are the players and their associated revenues are the bids. If the cost sharing mechanism decides to serve the current customer, then FairShare accepts that customer and otherwise rejects. The cost sharing methods we consider satisfy several nice properties such as *competitiveness*, *cost recovery*, and *cross-monotonicity.* Competitiveness implies that the sum of the cost shares assigned to the selected players is no more than the total cost of service for those players, while cost recovery implies that the sum of the cost shares covers a large fraction of the actual cost to serve the accepted players. Cross-monotonicity means that the cost shares of each player decreases as more players receive service. These three properties together imply that cost sharing methods provide a very good way to *share* the costs of production *fairly* across customers so that good selection decisions can be made. Leveraging these properties and previous results on the social welfare of cost sharing mechanisms, we are able analyze the performance of FairShare. We believe that our use of game theory mechanisms in an online optimization setting has potential for many more applications.

2.1.2 Literature Review

Supply chain models with customer selection have recently gained interest (for a broader literature on order acceptance problems, see the survey of Slotnick [80]). Xu et al. [93] considers a stochastic periodic-review inventory model with lead time and the possibility of sales rejection. Bhaskaran et al. [8] considers an inventory model with convex cost structure and the ability to backlog or refuse demand. Charikar et al. [20] provide an approximation algorithm for the facility location problem with offline customer selection. Some models considered offline *market* selection, where a market is a sequence or collection of demands requested by a customer over time (or in multiple locations) that must be either fully accepted or rejected. Van den Heuvel et al. [86] show that it is NP-hard to approximate the profit maximization variants of these models within any constant. This motivates the focus on cost minimization variants that we and others have studied. Geunes et al. [39] develop a general linear programming rounding framework to approximately solve several supply chain problems with offline market selection and stochastic demand. Their framework gives constant factor approximations to the ELS, JR, and FL problems with offline market selection. In contrast, in this paper we consider markets that consist only of a single period (or location), but the market/customer selection decisions are made online with *no* information about the future. Earlier work on the offline prize-collecting traveling salesman problem and the prize-collecting Steiner tree problem studied by Bienstock et al. [11] and Goemans and Williamson [40], respectively, also falls into our customer selection framework.

There has also been a stream of literature on customer selection in the context of admission control problems. For example, Carr and Duenyas [17] and Caldentey and Wein [16] both consider a make-to-stock queue that is committed to serving long term contracts, but also has a spot market from which it needs to accept and reject orders. In Plambeck et al. [70] and Gallien et al. [36], there is a queue that admits customers based on their revenue and amount of capacity available for their order. The major difference between this literature and our work is that we are not inherently limited by a capacity, which is the major factor for rejecting customers in these models. Instead, in our work customers are linked together due to the economies of scale in production, and a customer may be rejected based on his own projected profitability.

Only recently has the concept of online optimization been used to study models in operations management, although thus far only to problems without market selection. In Van den Heuvel and Wagelmans [85], the competitive ratio of the online economic lot sizing problem is shown to have a lower bound of two, matching the best known guarantee achieved in Axsater [2]. More general single item models are considered in Wagner [90]. In Buchbinder et al. [14], an online primal-dual algorithm is proposed for a make-to-order variant of the online joint replenishment problem which has a competitive ratio of three. Fotakis [34] provides a lower and matching upper bound depending on the number of customers for the online facility location problem. The work of Meyerson [62] provides a constant competitive algorithm for a variant of the online facility location problem with a mild assumption on the customer arrivals. In Ball and Queyranne [4], booking policies are found that achieve small competitive ratios for online revenue management problems. In Keskinocak et al. [53], online algorithms are developed for scheduling problems with lead time quotations. Finally, Jaillet and Lu [47] propose algorithms for the online traveling salesman problem with online customer selection. The ideas used in all these papers are very different than the ones used in this work.

The area of cooperative game theory, which focuses on how to fairly allocate costs among players, has gained recent interest in operations management (see Cachon [15], Nagarajan and Sosic [65], Bhaskaran and Krishnan [9], Kim and Netessine [55]). Of main interest to this paper are the cost sharing mechanisms for inventory management (Xu and Yang [91]), facility location (Pál and Tardos [68]) and Steiner tree (Jain and Vazirani [48]) problems. Roughgarden and Sundararajan [73] developed a notion called *summability*, which was used to study the social welfare of the previously mentioned mechanisms. Our paper is the first to leverage cost sharing mechanisms directly in an online optimization setting.

2.2 General Model

Next we describe the details of the core model, and discuss different extensions and applications in Section 2.2.1. The models studied in this paper involve decisions that are made in two phases. First, there is a *selection phase* in which customers arrive sequentially in an online manner. In *stage* k of the selection phase, customer k arrives with requirements I_k and rejection cost r_k. Requirements (information) I_k may include demand quantities, due dates, and locations needed by the customer. After customer k arrives, the supplier needs to decide whether to accept or reject customer k using only the information regarding the first k customers. If customer k is accepted, then he must be served according to his requirements during the production phase. If customer k is rejected, a cost of r_k is incurred, which may represent lost revenue or a price paid to a third-party supplier. Since the cost of each customer is not necessarily separable due to the economies of scale in production, evaluating the marginal cost of a customer a priori is challenging and thus makes the selection problem nontrivial. The selection phase completes when the supplier stops observing new customers. At this point the customers that arrived have been partitioned into accepted and rejected sets denoted by $\mathcal{A}$ and $\mathcal{R}$, respectively.

The second phase is the *production phase*, where the accepted customers are served accordingly to meet their requirements. Let $\mathcal{Q}$ be the set of production options that are available to the supplier. The production options may represent potential order dates, locations of facilities, or just a single production setting. For a nonempty set of production options $Q \subseteq \mathcal{Q}$ and a set of customers T, $P(Q, T)$ denotes the minimum possible production cost to serve the customers in T using only the production options in Q. The function $P(Q, T)$ typically represents the cost of an optimal solution to a minimization problem. The production cost for the accepted set of customers $\mathcal{A}$ is then denoted by $P(\mathcal{Q}, \mathcal{A})$, which implies that the supplier could potentially use all options available to serve the accepted customers. We make the natural assumptions that $P(\mathcal{Q}, T)$ is nondecreasing in T and $P(\mathcal{Q}, \emptyset) = 0$. The overall goal is to minimize the total production costs of the accepted customers plus the rejection costs of the rejected customers. This two phase problem is generally referred to as an *online customer selection* problem. In Section 3 we will describe specific applications of the model by specifying definitions for I_k, r_k, $\mathcal{Q}$, and $P(\cdot, \cdot)$.

We now outline convenient notation used throughout the paper. Let N be the number of customers that arrived (unknown a priori), U be the full set of customers $\{1, \dots, N\}$ and U_k be the first k customers $\{1, \dots, k\}$, implying that $U = U_N$. When referring to the final stage N, the subscript may be dropped for simplicity. The rejection cost of a subset of customers $T \subseteq U$ is defined as $R(T)$, i.e., $R(T) = \sum_{k \in T} r_k$. The notation $\mathcal{A}_k$ and $\mathcal{R}_k$ denote the customers that were accepted and rejected, respectively, by the online algorithm through the first k stages. Note that $\mathcal{A}_k \cup \mathcal{R}_k = U_k$, $\mathcal{A}_k \cap \mathcal{R}_k = \emptyset$, $\mathcal{A}_{k-1} \subseteq \mathcal{A}_k$, and $\mathcal{R}_{k-1} \subseteq \mathcal{R}_k$ for all k.

If all information $I_1, \dots, I_N$ and $r_1, \dots, r_N$ is known upfront, then the *offline customer selection problem* is defined as $\mathrm{OPT}(\mathcal{Q}, U) = \min_{A \subseteq U} \ P(\mathcal{Q}, A) + R(U \backslash A)$. Let $\mathcal{A}_k^*$ and $\mathcal{R}_k^*$ denote an optimal pair of accepted and rejected sets in the offline problem $\mathrm{OPT}(\mathcal{Q}, U_k)$. Note that $\mathcal{A}_k^* \cup \mathcal{R}_k^* = U_k$ and $\mathcal{A}_k^* \cap \mathcal{R}_k^* = \emptyset$ for all k, but monotonicity does not necessarily hold since an offline solution may change its selection decisions as the stages progress. If there are multiple optimal solutions, we will assume that $\mathcal{A}_k^*$ is a maximal one, which means it is not contained in another optimal set of accepted customers.

The optimal offline cost through stage k is denoted by $C^*(U_k)$, which can be expressed as $C^*(U_k) = P(\mathcal{Q}, \mathcal{A}_k^*) + R(\mathcal{R}_k^*)$. The value $C(U_k)$ denotes the total cost incurred by the online algorithm through stage k, i.e., $C(U_k) = P(\mathcal{Q}, \mathcal{A}_k) + R(\mathcal{R}_k)$. Using these definitions, it follows that for any online algorithm and any stage k, $C^*(U_k) \leq C(U_k)$. Finally, we will sometimes drop the production options input from P and OPT which implies that we are using the entire set $\mathcal{Q}$, i.e., $P(\mathcal{Q}, \cdot) = P(\cdot)$ and $\mathrm{OPT}(\mathcal{Q}, \cdot) = \mathrm{OPT}(\cdot)$.

The performance of an online algorithm is evaluated using the notion of *competitive ratio*. An algorithm has a competitive ratio of α and is called α-competitive if $C(U) \leq \alpha C^*(U)$ for any online sequence of customers U and their respective characteristics. In other words, the cost of the algorithm is guaranteed to be at most α times the cost of an optimal offline solution for *any* customer sequence.

2.2.1 Model Applications

Traditionally, customer selection decision are made based on available-to-promise strategies, which basically attempt to determine the feasibility of an order based on available inventory or capacity. In our framework, we implicitly assume that the order has already passed a feasibility check, and we now are interested in whether the order will be profitable. For clarity purposes, the core model described only has two phases. However, our model and results can be easily extended to the scenario where there are multiple consecutive phases of selection and production, as long as the customer orders corresponding to a certain production phase arrive before that phase begins. For example, if each month is considered to be a distinct production phase, the corresponding selection phase is any time before that month begins. If customer lead times are at least a month, which is very common in many applications, then when every month begins the exact demand is known and a minimum cost production plan can be created for that month. In addition, if we can defer selection decisions, then this actually makes the problem easier since it provides greater flexibility for the decision maker.

Specifically, the core model described above captures various settings, in which (i) customer order selection is a common practice or suppliers have the flexibility to satisfy customer orders from either internal resources or spot markets/third parties; (ii) customer 'patience' is short (not necessarily immediate) relative to the typical requested/acceptable lead time to satisfy customer requests; and (iii) the underlying supply/production cost structure is characterized by economies of scale that make the selection decisions for different customer orders highly dependent. Under this type of cost structure, production is typically planned well in advance.

The steel, glass and construction industries are examples that fit the characteristics described above. For example, consider a supplier that sells construction materials. When customers, i.e., contractors, request products with specific due dates, the supplier may decide that the quantity ordered does not cover the costs of materials and delivery. Note that this decision might depend on the requests of previous customers since their orders might have already covered some fixed costs. Since the contractors have deadlines to complete their projects, an instantaneous, i.e., 'online', decision is typically required to allow them to plan accordingly. Typically the supplier will communicate to the customer that they are out of stock as a way to reject the customer if necessary. The production cost functions for this type of application can be modeled by submodular or inventory control problems which we discuss in Sections 2.3.1, 2.3.2, and 2.3.3.

The model also captures typical scenarios in the service industry. For example, consider a service provider that serves customers through a physical infrastructure. Typically, customers will request service from the provider and the provider will need to decide whether to serve each customer or not. In order to serve a customer, the proper infrastructure or facility must be setup near the customer. If the infrastructure needs to be improved or expanded at a relatively large cost for the customer to receive service, then he might be rejected by the provider ("out of network"). Otherwise, the customer is accepted and incurs a cost corresponding to his usage of the service. The production cost function in this setting could be modeled with the facility location problem or network design problems in Sections 2.3.4 and 2.3.5.

Other scenarios that can be captured by our models are when the 'rejection' cost represents a fee paid to a third-party supplier/logistics company or the use of spot markets (versus internal capacity) to satisfy the customer order. This situation can arise when the cost of serving customers using external resources is sometimes cost effective. For example, a local company may prefer to do some bulk shipping of their own but may rely on national companies to ship smaller or more long-distance packages. The use of third party logistics (3PL) is becoming increasingly commonplace and operationally advantageous.

The model can also be used as a tool to enhance *available-to-promise* strategies, in which companies interact with customers regarding their desired orders and delivery dates. Our model can be used as a support tool for negotiating with customers. For example, rather than rejecting a customer, a supplier may ask the customer to modify his requirements in order to create a mutually beneficial deal. This may include increasing the demand quantity, being more flexible with the due date, and/or reducing the variety of the order. Specifically, one could increase the demand quantity until the customer selection algorithm accepts the order in order to find a new deal to counteroffer to the customer. Alternatively, given the demand information of the customer, one can find all the due dates for which the customer selection algorithm would lead to accepting that order, and then present these options to the customer.

In other important settings, where the lead time of customers is short relative to the production phase, the selection phase and the corresponding production phase could overlap. The online customer selection model can easily be applied to these situations, but one would need to use, in addition to the online selection algorithm, an online production algorithm to solve the production problem since not all the accepted customers are known in advance. Although these problems are important, they are also much more complex. In fact, for many of the models discussed in this paper, just the online production problem with no customer selection is hard in the sense that there is no online algorithm with a constant competitive ratio.

2.3 Examples

2.3.1 Submodular Cost Problems with Online Customer Selection

In this section, we consider the case where the production cost function is submodular and there is only one production option, i.e., $|\mathcal{Q}| = 1$. Furthermore, the rejection cost for each customer k, denoted by r_k, can be an arbitrary nonnegative number independent of his requirements, I_k. A function $P(\cdot)$ is *submodular* if for all $S \subseteq T \subseteq$

U and $i \notin T$, $P(S \cup \{i\}) - P(S) \geq P(T \cup \{i\}) - P(T)$. Equivalently, a function $P(\cdot)$ is submodular if for all $S, T \subseteq U$, $P(S) + P(T) \geq P(S \cap T) + P(S \cup T)$.

Submodular functions with online customer selection are simply online customer selection problems where the production cost function $P(\cdot)$ is submodular. Nondecreasing submodular functions arise naturally in many applications where there are economies of scale. We provide several examples below, all of which naturally have $|\mathcal{Q}| = 1$.

Example 1 (To Build or Not to Build)**.** *Consider the function $P(T) = K$ if $|T| > 0$, and $P(\emptyset) = 0$. This function essentially builds or implements a project if there is any customer that needs to be served. When a customer k arrives online, his willingness to pay, r_k, is revealed. Due to the simplicity of $P(\cdot)$, I_k has no particular meaning.*

Example 2 (Multicast Routing)**.** *Consider a tree $\mathcal{T}$ with root v. Each edge $e_j \in \mathcal{T}$ has a cost $c_j \geq 0$. Let T be a subset of nodes and $P(T)$ be the cost of connecting the nodes in T to the root v using only edges in $\mathcal{T}$. Then $P(T)$ is clearly nondecreasing and submodular and is referred to as the multicast routing problem (Deering and Cheriton [25]). The information I_k for customer k would be his node location, and r_k is his willingness to pay for service from that node.*

Example 7 in Appendix A.1 describes another application regarding polymatroid optimization and a continuous inventory replenishment problem.

2.3.2 Economic Lot Sizing Problem with Online Customer Selection

The *Economic Lot Sizing (ELS)* problem is a single item, discrete time inventory model. There is a set of customers, each with a due date and demand quantity, that need to be served by a sequence of production orders over a planning horizon of a fixed number of periods. Each order incurs a setup cost K. Each customer can only be served by an order prior to his due date. If a customer with due date t and demand d is satisfied by an order at time $s < t$, then a per unit holding cost

$h > 0$ is incurred for every period and every unit of demand carried in inventory from period s to t. Without loss of generality we can assume that each customer is served from the latest order prior to his due date. The objective is to minimize the total setup ordering cost plus holding cost. The ELS problem can be solved efficiently via dynamic programming (Wagner and Whitin [89]).

If the supplier has an option to not serve customers, or to subcontract customers to a third party, then we say the supplier can reject customers (or select only some of the customers). Let r_k be the rejection cost of customer k. In this full information model, the goal is to decide which customers to select and how to serve these customers. The objective is to minimize the rejection cost of the rejected customers plus the ELS setup and holding costs to satisfy the the selected customers. This is called the *Economic Lot Sizing problem with Offline Customer Selection*, which can be solved efficiently (Geunes et al. [37]).

We will focus on the *Economic Lot Sizing problem with Online Customer Selection.* Customers arrive in an online manner and the supplier needs to make an immediate selection decision before new customers arrive. When a customer k arrives, he specifies $I_k = (d_k, t_k)$, where d_k is the quantity and t_k is the due date. If the rejection cost per unit is fixed at r, then the rejection cost of customer k is then $r_k = rd_k$. (For convenience, we assume r is an integer multiple of h.) The set of production options $\mathcal{Q}$ is the set of possible order dates. The production cost $P(Q, T)$ is then the optimal cost of the ELS problem on a subset of customers T using only the potential order dates $Q \subseteq \mathcal{Q}$.

2.3.3 Joint Replenishment Problem with Online Customer Selection

The *Joint Replenishment (JR)* problem is a natural extension of the ELS problem with multiple item types, indexed $1, \ldots, M$. The goal is to serve a set of customers, each with a quantity, due date, and item type, by a sequence of production orders over a planning horizon of a fixed number of periods. Each order incurs a joint setup cost of K_0. In addition, for each item type i ordered, an item setup cost of K_i is incurred. Each customer can only be served by orders that contain his item type and are before his due date. As with the ELS problem, there is also a per unit holding cost h. The objective is to minimize the total setup ordering cost plus holding cost. This problem was shown to be NP-hard in Arkin et al. [1], and admits a current best approximation guarantee of 1.80 due to Levi et al. [58]. More general JR problems are considered in Cheung et al. [23].

When the supplier does not have to serve all the customers, but can now reject customers at a per unit cost r_k, then the problem becomes even more complex. The supplier must now decide on the optimal set of customers to select and decide how to serve them. Specifically, the goal is to minimize the total rejection costs plus the cost of the JR problem on the accepted customers. This is called the *Joint Replenishment Problem with Offline Customer Selection.* This problem was first studied in Geunes et al. [39] who gave a 2.35-approximation for the market selection variant.

In this work, we focus on the *Joint Replenishment Problem with Online Customer Selection.* Like the previous models, customers arrive one after the other and must be either accepted or rejected immediately. Specifically, when a customer k arrives, his requirements $I_k = (d_k, t_k, i_k)$ are observed, where d_k specifies the quantity, t_k specifies the due date, and i_k specifies the item type. If the rejection cost per unit is fixed at r, then the rejection cost for customer k is then $r_k = rd_k$. (For convenience, we assume r is an integer multiple of h.) We again define the set of production options $\mathcal{Q}$ as the set of potential order dates. The production cost function $P(Q, T)$ is now the optimal cost of the JR problem with production options Q and customers T.

2.3.4 Facility Location Problem with Online Customer Selection

The metric *Facility Location (FL)* problem is a well studied NP-hard problem. The goal is to serve a set of customers, each with a specified demand quantity and location, by opening a set of facilities. There are M potential facilities, indexed by

$j = 1, \ldots, M$. The opening cost of facility j is f_j. Each customer k is served by the nearest open facility, and pays a service cost $c(j,k)$ per unit if served by facility j. The assumption is that $c(\cdot,\cdot)$ induces a metric (symmetric and satisfies triangle inequality) over the facilities and customers. The goal is to serve all the customers so as to minimize the total facility costs plus service costs. The first constant factor approximation algorithm was given by Shmoys et al. [79], and the current best approximation factor is 1.488 due to Li [60].

When the supplier does not have to serve all the customers, but can now reject customers at a cost r_k, then the problem becomes even more complex. The supplier must now decide on the optimal set of customers to accept and decide how to serve them. Specifically, the goal is to minimize the total rejection costs plus the cost of the FL problem on the accepted customers. This is called the *Facility Location Problem with Offline Customer Selection.* The first approximation algorithm was given by Charikar et al. [20] (who call this FL with outliers), and the current best approximation factor of 1.85 is due to Xu and Xu [92]. Geunes et al. [39] gives a 2.06-approximation algorithm for the market selection variant of this problem.

Here we focus on the *Facility Location Problem with Online Customer Selection.* When a customer k arrives online, we observe $I_k = (d_k, l_k)$, where d_k specifies the quantity and l_k specifies the location. If the rejection cost per unit is fixed at r, the the rejection cost for customer k is then $r_k = rd_k$. Let $\mathcal{Q}$ denote the potential set of facilities. The production cost function $P(Q,T)$ is now the optimal cost of the FL problem for a given set of customers T that can only use the facilities in Q.

2.3.5 Network Design Problems with Online Customer Selection

The *Steiner tree* problem is a network design model where we are a given a graph $G = (V,E)$ and subset of nodes $S \subset V$ that needs to be connected with a tree. The cost of the tree is the sum of the cost of each edge used in the tree. Extensions of this problem include the Steiner forest problem (need to connect set of node pairs), the Single Source Rent-or- Buy (SSROB) problem (connect nodes + each edge either rented per use or bought for fixed price), and the Multi-commodity Rent or Buy (MROB) problem (connect node pairs + each edge either rented per use or bought for fixed price).

When nodes (node pairs) can be rejected, these are typically referred to as *prize-collecting* problems. We focus on the *Network Design Problems with Online Customer Selection* where customers arrive online and the information I_k for customer k is his node location/pair and rejection cost r_k. The production cost $P(T)$ is the optimal cost of the network design problem to serve the customers in T. (We do not need to use the notion of production options for this example.)

2.4 Results

There are two main set of results in this paper, depending on the structure of the rejection costs. If the rejection cost per unit is fixed, then Copycat and StablePair perform quite well, as seen in Table 1 below. The first column denotes the problem, the second and third columns denote the competitive ratio guarantees we can get with Copycat and StablePair respectively, and the fourth column is a lower bound on the best competitive ratio. In Sections 2.5 and 2.6, we will show general theorems for the Copycat and StablePair Algorithms. In Section 2.7, we will demonstrate how to exactly get the competitive ratio guarantees for the submodular, ELS, FL, and JR problems with online customer selection. Note that the submodular results also hold for arbitrary per unit rejection costs! In Section 2.9.1, we will show how we obtained the lower bound of 2 for all the problems in Table 2.1.

Table 2.2 below summarizes the results using the FairShare algorithm, which we use for problems with aribitrary per-unit rejection costs. For simplicity, we now assume every customer requests one unit of demand. (All the results hold with multiple unit requests by replacing N with the total number of units demanded.) The first column denotes the production cost problem. The second column denotes the competitive ratio achieved by using FairShare. The third column shows lower

Table 2.1: Competitive ratio guarantees for problems with fixed per unit rejection costs.

Problem	Copycat	StablePair	Lower Bound
Submodular*	2	2	2
Economic Lot Sizing	3	3	2
Joint Replenishment	4	3	2
Facility Location	4	2.41	2

bounds on the best possible competitive ratio. In Section 2.8, we will show how to obtain the competitive ratios from FairShare. In Section 2.9.2 we will demonstrate how to obtain the lower bound for the FL problem with online customer selection.

Table 2.2: Results for problems with arbitrary per unit rejection costs.

Problem	FairShare	Lower Bound
Economic Lot Sizing	$O(\sqrt{\log N})$	$\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$
Facility Location	$O(\sqrt{\log N})$	$\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$
Network Design	$O(\log N)$	$\Omega(\sqrt{\log N})$
Machine Scheduling	$O(\sqrt{\log N})$	open

Note that these results rely on methods and results from other papers, *combined* with our analysis in Section 2.8. Those papers include [91], [68], [48], [56], [42], [74], [73], [21], and Brenner and Schäfer [13]. We note that our competitive ratio guarantee for the Steiner tree problem with online customer selection was also achieved in Qian and Williamson [71], however in there version they also had to solve the Steiner tree problem online as well. All of the other network design problems with online customer selection previously had no known results we are aware of.

2.5 Copycat Algorithm

We now provide a framework to solve problems with online customer selection which we call the *Copycat Algorithm*. Simply put, for every customer arrival k, the offline problem OPT(U_k) is solved to obtain an optimal offline solution $\mathcal{A}_k^*$. Then the arriving customer k is accepted if and only if customer k is accepted in the respective optimal solution (i.e., $k \in \mathcal{A}_k^*$). Otherwise, $k \in \mathcal{R}_k^*$ and customer k is rejected. This is called the Copycat Algorithm because it simply copies the optimal offline solution's decision at each customer arrival.

Copycat Algorithm: Accept current customer k if and only if $k \in \mathcal{A}_k^*$.

Although Copycat appears naive, it can be shown that it performs well for the class of problems considered in this paper. Note that fixing our previously made decisions when we re-optimize will result in most customers getting rejected and an overall poor performance. See Example 3 in Appendix A.1 for a detailed example. We next show a surprising property of this algorithm that only requires the monotonicity of $P(\cdot)$. Specifically, the next lemma asserts that the rejection cost for Copycat will never be too large for *any* problem with online customer selection.

Lemma 1. *Assume that $P(\cdot)$ is nondecreasing. Then the total rejection costs of the Copycat Algorithm at each stage k is at most the respective optimal offline cost, i.e., $R(\mathcal{R}_k) \leq R(\mathcal{R}_k^*) + P(\mathcal{A}_k^*) = C^*(U_k)$ for all k. Specifically, the final rejection cost $R(\mathcal{R}) \leq C^*(U)$.*

Proof. The proof is by induction. Start with the base case $k = 1$, where

$$R(\mathcal{R}_1) = R(\mathcal{R}_1^*) \leq R(\mathcal{R}_1^*) + P(\mathcal{A}_1^*) = C^*(U_1).$$

The first equality follows from the fact that $\mathcal{R}_1 = \mathcal{R}_1^*$ since the selection decision made by OPT(U_1) is copied. The inequality follows from the nonnegativity of $P(\cdot)$. The last equality follows directly from the definition of $C^*(\cdot)$.

Now assume the inductive hypothesis that $R(\mathcal{R}_{k-1}) \leq C^*(U_{k-1})$. Consider the following two cases depending on whether customer k was accepted or rejected in the solution of OPT(U_k).

Case 1) If customer k is accepted in the solution of OPT(U_k) (i.e., $k \in \mathcal{A}_k^*$), then

$$R(\mathcal{R}_k) = R(\mathcal{R}_{k-1}) \le C^*(U_{k-1}) \le C^*(U_k).$$

The first equality holds because $k \in \mathcal{A}_k^*$ which implies that Copycat accepted k, and thus $\mathcal{R}_k = \mathcal{R}_{k-1}$. The inequality follows from the inductive hypothesis. The last inequality follows from the fact that any solution to OPT(U_k) induces a solution to OPT(U_{k-1}) with cost at least $C^*(U_{k-1})$.

Case 2) If customer k is rejected in the solution of OPT(U_k) (i.e., $k \in \mathcal{R}_k^*$), then

$$R(\mathcal{R}_k) = R(\mathcal{R}_{k-1}) + r_k \le C^*(U_{k-1}) + r_k = C^*(U_k).$$

The first equality holds since $k \in \mathcal{R}_k^*$ which implies that Copycat also rejected k. The inequality follows from the inductive hypothesis. The last equality holds by the linearity of $R(\cdot)$ and the fact that there is an optimal solution to OPT(U_k) that rejected customer k. $\square$

Using Lemma 1, it is clear that if the production costs of Copycat are no more than β times the optimal offline cost, a competitive ratio of $\beta + 1$ is obtained. This is stated precisely in the following theorem.

Theorem 1. *Let $\mathcal{A}$ be all the customers that the Copycat Algorithm accepts and let β be a positive scalar. If $P(\mathcal{A}) \le \beta C^*(U)$, then Copycat is $(\beta+1)$-competitive.*

Interestingly, we can also show that the Copycat Algorithm works well if the sequence of optimal solutions satisfies a certain property. Specifically, if satisfying $\cup_{i=1}^k \mathcal{A}_i^*$ has cost at most $\beta C^*(U_k)$, then the same holds for satisfying $\mathcal{A}$ since $\mathcal{A} \subseteq \cup_{i=1}^k \mathcal{A}_i^*$ by definition of the Copycat Algorithm. Combining this fact with Lemma 1, we obtain the following lemma.

Lemma 2. *If $P(\cup_{i=1}^k \mathcal{A}_i^*) \le \beta C^*(U_k)$, then the Copycat Algorithm is*
$(\beta + 1)$- competitive.

In the subsequent sections, we shall show how to obtain bounds like the one in Theorem 1 above for several interesting problems and cases. However, one potential major flaw with the Copycat Algorithm is that it requires an exact solution to the offline problem in each stage. Even worse, the offline problem may sometimes be NP-hard such as in the JR and FL applications described in Sections 2.3.3 and 2.3.4, respectively. Motivated by these issues, we present another algorithm in the next subsection that is efficiently computable these applications.

2.6 StablePair Algorithm

We now provide another algorithm for online customer selection called the *StablePair Algorithm*. For a nonempty subset of production options $Q \subseteq \mathcal{Q}$ and customers $T \subseteq U$, we call (Q, T) a *stable pair* if there exists an optimal solution to the respective offline customer selection problem defined on Q and T, denoted by $\mathrm{OPT}(Q, T)$, that accepts all of the customers in T. The StablePair Algorithm accepts a given customer k if and only if there exists a stable pair $(Q, T) \subseteq (\mathcal{Q}, U_k)$, such that $k \in T$.

StablePair Algorithm: Accept current customer k if and only if there exists a stable pair $(Q, T) \subseteq (\mathcal{Q}, U_k)$ such that $k \in T$.

As we shall show, the StablePair Algorithm has two main benefits. First, a stronger bound on the rejection costs can be obtained, and second, the selection phase can be implemented in polynomial time for many interesting problems that we consider. The stability name arises from the fact that if a customer was accepted by StablePair, then he would also be accepted if he had arrived at a later date. (This is not necessarily true for the Copycat Algorithm.) Moreover, in the next lemma it is shown that the StablePair Algorithm is less conservative than the Copycat Algorithm in that each customer accepted by Copycat is also accepted by StablePair. (We let the superscripts C and S refer to the Copycat and StablePair Algorithms decisions, respectively.)

Lemma 3. *The accepted set of customers by StablePair is at least that of Copycat, i.e.,* $\mathcal{A}^C \subseteq \mathcal{A}^S$.

Proof. Consider a customer $k \in \mathcal{A}^C$ and let $\mathcal{A}_k^*$ be the optimal solution induced by $\mathrm{OPT}(\mathcal{Q}, U_k)$. Since k was accepted by the Copycat Algorithm, then $k \in \mathcal{A}_k^*$. This implies that $(\mathcal{Q}, \mathcal{A}_k^*)$ is a stable pair for k since there is a solution to $\mathrm{OPT}(\mathcal{Q}, \mathcal{A}_k^*)$ that accepts all of $\mathcal{A}_k^*$. Therefore $k \in \mathcal{A}^S$ and $\mathcal{A}^C \subseteq \mathcal{A}^S$. □

We now give another useful way of characterizing a stable pair which follows immediately from the definition. The pair (Q, T) is stable if and only if

$$R(S) \geq P(Q, T) - P(Q, T \backslash S) \qquad \forall\, S \subseteq T. \tag{2.1}$$

Note that (2.1) above is equivalent to not having any solution to the offline problem defined on Q and T that is strictly better than accepting T. As already mentioned, the StablePair Algorithm achieves a stronger bound for the rejection costs $R(\mathcal{R})$. Specifically, compared to Lemma 1, in Lemma 4 below the term $R(\mathcal{A} \cap \mathcal{R}^*)$ is no longer needed in the bound, which will later lead to better overall competitive ratios than Copycat.

Lemma 4. *Assume that* $P(\cdot)$ *is nondecreasing. The StablePair Algorithm for online customer selection problems has rejection cost* $R(\mathcal{R}) \leq R(\mathcal{R} \cap \mathcal{R}^*) + P(\mathcal{R} \cap \mathcal{A}^*) \leq R(\mathcal{R} \cap \mathcal{R}^*) + P(\mathcal{A}^*)$*, for any offline optimal solution* $(\mathcal{A}^*, \mathcal{R}^*)$.

Proof. We first show that for all $X \subseteq \mathcal{R}$, $R(X) \leq P(X)$. Let $X \subseteq \mathcal{R}$ and assume for contradiction that

$$R(X) > P(\mathcal{Q}, X) = P(X). \tag{2.2}$$

Let k be the last arriving customer in X. Since k was rejected by StablePair, it follows that $(\mathcal{Q}, X)$ is not a stable pair for customer k. Thus, by Eq. (2.1) there exists a set $S \subseteq X$ such that

$$R(S) < P(\mathcal{Q}, X) - P(\mathcal{Q}, X \backslash S) = P(X) - P(X \backslash S). \tag{2.3}$$

Note that $S \neq \emptyset$ or else $0 < 0$. Subtracting (2.3) from (2.2) yields

$$R(X \backslash S) > P(X \backslash S). \tag{2.4}$$

Now reset $X \leftarrow X \backslash S$ and repeat the analysis above until $X = \emptyset$. This will eventually give a contradiction that $0 > 0$. Now let $(\mathcal{A}^*, \mathcal{R}^*)$ be any optimal offline solution and let $X = \mathcal{R} \cap \mathcal{A}^*$. From the previous argument, it follows that

$$R(\mathcal{R} \cap \mathcal{A}^*) \leq P(\mathcal{R} \cap \mathcal{A}^*). \tag{2.5}$$

Adding $R(\mathcal{R} \cap \mathcal{R}^*)$ to both sides of (2.5) completes the proof. □

Copycat and StablePair are different in at least three ways. First, Example 4 in Appendix A.1 demonstrates that the Copycat Algorithm is indeed strictly weaker with respect to bounding the rejection costs. Second, Example 5 in Appendix A.1 shows that Copycat can "regret" accepting customers, where as StablePair never has regret since once a customer is accepted, there is always a stable pair (i.e., the original one) that would accept him later on. Finally, Example 6 in Appendix A.1 shows that StablePair can accept customers that would never be accepted by $\mathcal{A}_k^*$ for any k.

Using Lemma 4 (which holds for any optimal solution), it is clear that if we can obtain a bound on the production costs of StablePair, then we can obtain strong competitive ratios that can be strictly better than Copycat. This is made precise in the following theorem.

Theorem 2. *Let $\mathcal{A}$ be all the customers that the StablePair Algorithm accepts and let β and γ be positive scalars. If there exists an optimal offline solution $(\mathcal{A}^*, \mathcal{R}^*)$ such that $P(\mathcal{A}) \leq \beta P(\mathcal{A}^*) + \gamma R(\mathcal{R}^*) + R(\mathcal{A} \cap \mathcal{R}^*)$, then StablePair is* $\max(\beta + 1, \gamma + 1)$*-competitive.*

Since Theorem 2 can be used by showing the bound for just one optimal solution, for the rest of the paper we will always assume that we are using a solution $\mathcal{A}_k^*$ that is maximal. This means that no optimal set of accepted customers to $\mathrm{OPT}(U_k)$ is a strict superset of $\mathcal{A}_k^*$.

Although the online customer selection decisions can be done efficiently with StablePair, the decision maker still needs to calculate $P(\mathcal{A})$ in the production phase, which might be NP-hard to solve. In most practical settings, $P(\mathcal{A})$ can be calculated via integer programming or other methods. Indeed, if one desires to solve the production phase using a c-approximation algorithm (an algorithm that finds a solution in polynomial time that is no more than c times the optimal cost), then the theoretical competitive ratio would simply increase multiplicatively with c. The following theorem makes this statement precise and follows directly from the definition of an approximation algorithm and Lemma 4.

Theorem 3. *Let $\mathcal{A}$ be the customers that StablePair accepted and let β, γ, and $c > 1$ be positive scalars. Assume there exists an optimal offline solution $(\mathcal{A}^*, \mathcal{R}^*)$ such that the optimal cost of serving $\mathcal{A}$ is $P(\mathcal{A}) \leq \beta P(\mathcal{A}^*) + \gamma R(\mathcal{R}^*) + R(\mathcal{A} \cap \mathcal{R}^*)$. Then the StablePair Algorithm is* $\max(c\beta + 1, c(\gamma + 1))$*-competitive when a c-approximation algorithm for $P(\cdot)$ is used to serve $\mathcal{A}$.*

Note that the previous theorem also holds if we use a c-competitive online algorithm for solving $P(\mathcal{A})$. This may be useful if one wishes to merge the online customer selection and production phases together, which we do in Section 2.7.3. The remainder of this paper focuses on how to implement the StablePair Algorithm efficiently and how to bound the production costs of StablePair for several inventory and logistics problems. Since $\mathcal{A}^C \subseteq \mathcal{A}^S$ and $P(\cdot)$ is nondecreasing, then these production bounds will hold for the Copycat Algorithm as well. Not surprisingly, the production bounds we obtain are highly dependent on the combinatorial structure of the production problems.

2.6.1 StablePair Implementation

Submodular Implementation

The proof of Lemma 8 below will imply that the Copycat and StablePair Algorithms are identical if Copycat always outputs the maximal optimal solution. This trivially happens if there is always a unique optimal solution, which can be achieved via a random perturbation. Interestingly enough, it turns out that the Copycat Algorithm can be implemented efficiently for submodular production problems with online customer selection. For any set of customers T,

$$\mathrm{OPT}(T) = \min_{A\subseteq T}\ P(A) + R(T\backslash A) = \min_{A\subseteq T}\ P(A) + R(T) - R(A) = \min_{A\subseteq T}\ P(A) - R(A).$$

Since $R(\cdot)$ is modular, then $P(\cdot) - R(\cdot)$ is also submodular. Therefore, $\mathrm{OPT}(T)$ is just the solution to a submodular minimization problem, which can be solved in polynomial time (McCormick [61]). Thus Copycat (and StablePair) can be implemented efficiently.

ELS Implementation

Although the Copycat Algorithm for the ELS problem with online customer selection can be implemented in polynomial time by using dynamic programming, we describe the StablePair implementation since it is faster and also serves as a foundation for how to implement StablePair for the JR variant. Lemma 5 below, proved in Appendix A.2, describes an exact method for implementing the StablePair Algorithm. The idea is that one can reduce the search space for stable pairs by only considering those with one production option (order date).

Lemma 5. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q, T) with $k \in T$ such that*

1. *Q consists of one order date, i.e. $Q = \{t\}$ for some time t.*

2. *T includes exactly all customers that can be served from t with at most r per unit in holding cost, i.e. $T = \{j \in U_k | t_j \in [t, t + r/h]\}$.*

3. *The rejection costs of T exceed the production costs of serving T from an order at t, i.e. $R(T) \geq K + \sum_{j\in T} h(t_j - t)d_j$.*

Note that without loss of generality, one only needs to consider the potential order dates that correspond to due dates of U_k (since zero-inventory ordering policies are optimal). Thus, the maximum number of candidate stable pairs is at most N. Since checking stability of each pair takes $O(N)$ time, then this characterization of the StablePair Algorithm provides an efficient implementation.

JR Implementation

Implementing Copycat problem for the JR problem with online customer selection requires solving an NP-hard problem. However, StablePair can be implemented efficiently in time polynomial in the number of customers and items. Lemma 6 below, proved in Appendix A.2, shows that a stable pair exists if and only if there is a stable pair using only one order date that satisfies the properties below.

Lemma 6. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q, T) with $k \in T$ such that*

1. *Q consists of one order date, i.e. $Q = \{t\}$ for some time t.*
2. *If T contains customers of type i, then it contains exactly all type i customers with due dates in $[t, t + r/h]$. Let $T^i = \{j \in U_k | t_j \in [t, t + r/h] \text{ and } i_j = i\}$ and let $\mathcal{I}$ be a subset of item types. This property is equivalent to the property that $T = \cup_{i \in \mathcal{I}} T^i$.*
3. *The set of item types in T are those who can pay for their item ordering cost plus the holding costs, i.e. $\mathcal{I} = \{i | R(T^i) \geq K_i + \sum_{j \in T^i} h(t_j - t)d_j\}$.*
4. *The total rejection costs of T are at least the production costs of serving T from t, i.e. $R(T) \geq K_0 + \sum_{i \in \mathcal{I}} \left(K_i + \sum_{j \in T^i} h(t_j - t)d_j\right)$.*

This lemma provides an efficient implementation of StablePair since the set of potential stable pairs is at most N, and checking the four properties for each pair takes $O(MN)$ time.

FL Implementation

Using the Copycat Algorithm for the FL problem with online customer selection will be inefficient since solving the offline problem is NP-hard. Implementing the

StablePair Algorithm, however, can be implemented efficiently in time polynomial in the number of customers and facilities. Lemma 7 below, proved in Appendix A.2, shows that a stable pair exists if and only if there is a stable pair with one facility with the properties below.

Lemma 7. *The StablePair Algorithm accepts customer k if and only if there exists a stable pair (Q, T) containing k such that*

1. *Q consists of one facility location, i.e. $Q = \{j\}$ for some facility j.*
2. *T are all customers that can be served from j using at most r in service cost per unit, i.e. $T = \{i \in U_k | c(i, j) \leq r\}$.*
3. *The rejection costs of T exceed the production costs of serving T from facility j, i.e. $R(T) \geq f_j + \sum_{i \in T} c(i, j) d_i$*

Since the number of possible stable pairs is at most M, and checking the conditions takes at most $O(N)$, then clearly this lemma provides an efficient way to implement the StablePair Algorithm.

2.7 Performance of Copycat and StablePair

2.7.1 Submodular Analysis with Arbitrary Rejection Costs

The following theorem bounds the production costs incurred by StablePair for all submodular problems with online customer selection. From Lemma 3, this bound holds for Copycat as well.

Lemma 8. *If the StablePair (or Copycat) Algorithm is used for submodular problems with online customer selection, then $P(\mathcal{A}_k) \leq P(\mathcal{A}_k^*)$ for all k.*

Proof. By the monotonicity of $P(\cdot)$, it is sufficient to show that $\mathcal{A}_k \subseteq \mathcal{A}_k^*$, for each k. Therefore, it is now sufficient to prove that the StablePair Algorithm accepts k if and only if $k \in \mathcal{A}_k^*$.

Clearly if $k \in \mathcal{A}_k^*$, then $(\mathcal{Q}, \mathcal{A}_k^*)$ is a stable pair that StablePair can use to accept k. Now assume that k was accepted by the StablePair Algorithm, and let $(\mathcal{Q}, T)$ be the stable pair that accepted k. Then

$$R(T \backslash \mathcal{A}_k^*) \geq P(T) - P(T \cap \mathcal{A}_k^*) \geq P(T \cup \mathcal{A}_k^*) - P(\mathcal{A}_k^*).$$

The first inequality follows from (2.1) which is a property of a stable pair. The second inequality follows from the submodularity of $P(\cdot)$. This implies that accepting $\mathcal{A}_k^* \cup T$ is at least as cheap as only accepting $\mathcal{A}_k^*$ and rejecting $T \backslash \mathcal{A}_k^*$. Since we chose $\mathcal{A}_k^*$ to be a maximal optimal solution, this implies that $T \subseteq \mathcal{A}_k^*$ and thus $k \in \mathcal{A}_k^*$. $\square$

Combining Theorems 1 and 2 and Lemma 8 obtains the following main result.

Theorem 4. *The Copycat and StablePair Algorithms for submodular problems with online customer selection is 2-competitive.*

In Theorem 12, we show that the competitive ratio for any deterministic online algorithm is at least two, and thus Copycat and StablePair achieve the best possible result.

2.7.2 ELS Analysis with Fixed Rejection Costs

In this section, we provide a bound on the total production costs incurred by the StablePair Algorithm when applied to the ELS problem with online customer selection. It is well known that the optimal solution for the ELS problem has a zero inventory ordering (ZIO) property (Wagner and Whitin [89]). This means that orders are placed only when the on-hand inventory level is zero. This implies that each order covers all demands with due dates between the order date and the due date of the last customer served by the order. We refer to this range of due dates as the respective *setup interval* of the order. We now prove two simple lemmas, proved in Appendix A.2, regarding the structure of an optimal offline solution.

Lemma 9. *Let $(Q, T) \subseteq (\mathcal{Q}, U)$ be a stable pair. Each setup interval induced by the ELS solution to $P(Q, T)$ must have a length of at most r/h periods.*

Lemma 10. *Let $(Q, T) \subseteq (\mathcal{Q}, U)$ be a stable pair and let $[a, b]$ be a setup interval from the ELS solution for $P(Q, T)$. Then the interval $[a, b]$ must contain the due date of a customer in $\mathcal{A}^*$. Therefore, $[a, b]$ must intersect a setup interval from the ELS solution for $P(\mathcal{A}^*)$.*

Using Lemmas 9 and 10, it is next shown that the StablePair Algorithm incurs production cost at most twice the optimal offline cost. From Lemma 3, this also holds for the Copycat Algorithm.

Lemma 11. *The production cost incurred by StablePair (Copycat) Algorithm for the ELS problem is within twice the optimal offline cost, specifically, $P(\mathcal{A}) \leq 2P(\mathcal{A}^*) + 2R(\mathcal{A} \cap \mathcal{R}^*) \leq 2C^*(U)$.*

Proof. We will explicitly construct a feasible production plan that serves all of the customers in $\mathcal{A}$ and show that its cost is at most twice the optimal offline cost $C^*(U)$. This will also clearly hold for the optimal (minimum cost) production plan to serve the customers in $\mathcal{A}$. Let $s_1, \ldots, s_m$ denote the order periods in the ELS solution to $P(\mathcal{A}^*)$, i.e., the optimal production plan for customers accepted by the optimal offline solution. Let K^* and H^* represent the setup costs and holding costs, respectively, induced by $P(\mathcal{A}^*)$. Consider a production plan that places orders at times $s_1, \ldots, s_m$ as well as $s_1 - r/h, \ldots, s_m - r/h$. The total cost of these orders is exactly $2K^*$. Now serve all customers in $\mathcal{A} \cap \mathcal{A}^*$ by the same order that they were served in the ELS solution for $P(\mathcal{A}^*)$. The resulting total holding costs for these customers is at most H^*. The only customers left to be served are those in $\mathcal{A} \cap \mathcal{R}^*$.

Next, consider each customer $k \in \mathcal{A} \cap \mathcal{R}^*$, i.e., a customer accepted by StablePair but not by the optimal offline solution. The claim is that there exists some optimal order s_j such that customer k's due date $t_k \in [s_j - r/h, s_j + 2r/h]$. (See Figure 2-1 for an example of the accepted customers near an optimal order s_j.) Now consider the stable pair (Q, T) that made StablePair accept customer k. Lemma 10 implies that the setup interval within the production plan for $P(Q, T)$ that contained k also contains some customer $k' \in \mathcal{A}^*$. Let s_j be the order from which customer k' is served in the production plan for $P(\mathcal{Q}, \mathcal{A}^*)$. By Lemma 9, the lengths of the setup intervals in

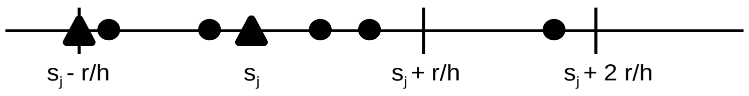

Figure 2-1: The triangles denote optimal orders and the circles denote due dates of customers. The first two customers will be served by the extra order placed at $s_j - r/h$. The next two customers are in $\mathcal{A}^*$ and are served by the order at s_j. The last customer will also be served by the order at s_j with holding cost at most $2r$ per unit.

any ELS solution are at most r/h. It then follows that $t_k \in [s_j - r/h, s_j + 2r/h]$. In the construction, customer k is served from the order $s_j - r/h$ if $t_k \in [s_j - r/h, s_j)$ and from s_j if $t_k \in [s_j, s_j + 2r/h]$. In either one of these cases, the per unit holding cost incurred by k is at most $2r$. However, customer k was rejected by the optimal offline solution and incurred a cost r. It follows that the total holding cost incurred by customers in $\mathcal{A} \cap \mathcal{R}^*$ is at most $2R(\mathcal{A} \cap \mathcal{R}^*)$. Summing up all the production costs yields an upper bound on $P(\mathcal{A})$ of $2K^* + H^* + 2R(\mathcal{A} \cap \mathcal{R}^*) \leq 2P(\mathcal{A}^*) + 2R(\mathcal{A} \cap \mathcal{R}^*) \leq 2C^*(U)$. □

Combining Theorems 1 and 2 with Lemma 11, we obtain the following main result.

Theorem 5. *The Copycat and StablePair Algorithms for the Economic Lot Sizing problem with Online Customer Selection are both 3-competitive.*

Example 8 in Appendix A.1 demonstrates that the analysis above is tight for both algorithms. We note that the analysis of the Copycat and StablePair Algorithms can be extended easily for non-decreasing setup costs K_t and more general holding cost structure. Specifically, if the cost of holding a unit from period s to t is h_{st}, then the analysis holds if h_{st} is subadditive (for any $s \leq u \leq t$, $h_{st} \leq h_{su} + h_{ut}$).

2.7.3 ELS Analysis Including Online Production

One extension of the ELS problem with online customer selection is when the online customer selection phase and production phase overlap. In this scenario, the supplier learns about each customer k at their arrival time e_k, which may be *after* the due dates of other customers. The information for each customer k is now $I_k = (e_k, d_k, t_k)$, where d_k and t_k are the respective demand quantity and due date for customer k. If customer k is accepted, he must be satisfied by an order in the time window $[e_k, t_k]$. Naturally we assume that $e_k \leq t_k$ and the arrival dates are chronological. The ELS problem with online customer selection and online production is more difficult, and in fact, Example 9 in Appendix A.1 demonstrates that we cannot obtain a constant competitive ratio for this problem, even when the optimal offline solution must also respect the arrival times.

However, if the due dates of the customers are in chronological order, then we can obtain a positive result as stated in the following theorem which is proved in Appendix A.2.

Theorem 6. *Consider the Economic Lot Sizing Problem with Online Customer Selection and Online Production. Assume that the due dates for the customers are in chronological order, i.e., $t_1 \leq t_2 \leq \ldots \leq t_N$. Then the Copycat Algorithm combined with a 2-competitive heuristic of Axsater [2] for the online ELS problem obtains an overall competitive ratio of 3.*

2.7.4 JR Analysis with Fixed Rejection Costs

In this subsection, we shall bound the production costs incurred by the StablePair Algorithm for the JR problem with online customer selection. For now, assume that there exists a black box to run the StablePair Algorithm. Again the setup interval of an order is defined as the range of due dates that the order serves, regardless of item type. Lemmas 9 and 10 both still hold with the same proofs for the JR problem with online customer selection. The following lemma describes the worst-case production costs that will be incurred.

Lemma 12. *If the StablePair (Copycat) Algorithm is used for the JR problem with online customer selection, then $P(\mathcal{A}) \leq 2P(\mathcal{A}^*) + R(R^*) + 2R(\mathcal{A} \cap \mathcal{R}^*) \leq 3C^*(U)$.*

Proof. The proof is similar in spirit to the proof of Lemma 11, but requires a more involved construction and analysis. Let $s_1, \ldots, s_m$ denote the times of the orders in the optimal production plan for $P(\mathcal{A}^*)$. The cost of the optimal production plan can be decomposed into the total setup costs, K^*, and the total holding costs, H^*. Let

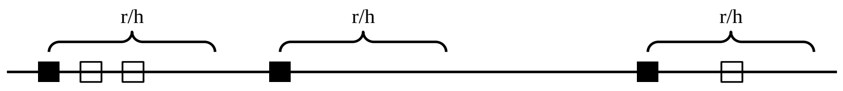

Figure 2-2: The six squares represent the order dates that make up $\hat{T}^i$. The solid squares represent the order dates that make up T^i and the hollow squares represent the dates that were pruned. By default, the first square must be solid. Then all squares within r/h are pruned and the process is repeated.

$\mathcal{A}^i$ simply denote the customers with item type i that are in $\mathcal{A}$. For each customer $k \in \mathcal{A}^i$, let a_k be the order date that serves customer k in the stable pair solution that StablePair used to accept customer k, and let $\hat{T}^i$ denote the set of these order dates sorted from earliest to latest. Construct $T^i \subseteq \hat{T}^i$ by processing $\hat{T}^i$ in order from earliest to latest, and remove any order date that is within r/h periods of the previous order date that was not removed. (See Figure 2-2 for an example of this construction.) Now consider the following two sets, X^i and Y^i. The set X^i is defined to be the set of all type i customers that have due date within $[t, t + r/h]$ for some $t \in T^i$. Note that X^i is not necessarily contained in the set $\mathcal{A}^i$. The set Y^i is then defined to be $\mathcal{A}^i \backslash X^i$.

To construct a solution that serves all the customers in $\mathcal{A}$, we first create the same sequence of orders as in $P(\mathcal{A}^*)$ in periods $s_1, \ldots, s_m$ and incur setups costs of K^*. Note that the item orders are replicated as well. All the customers in $\mathcal{A} \cap \mathcal{A}^*$ are then served in the the same way as in the production plan for $P(\mathcal{A}^*)$, and thereby incur holding costs of at most H^*. Now we create a sequence of duplicate orders shifted back by time r/h, i.e. at periods $s_1 - r/h, \ldots, s_m - r/h$, and incur another K^*.

Next, consider each item type i separately. Assume for now that for each order date $t \in T^i$, there exists an order s_j in the production plan for $P(\mathcal{A}^*)$, such that either $s_j \in [t - r/h, t]$ or $s_j - r/h \in [t - r/h, t]$. Moreover, order s_j (or $s_j - r/h$) includes item i. Assuming this property holds, one can bound the holding costs for the customers in $\mathcal{A}^i \cap \mathcal{R}^*$. Specifically, the property ensures that all type i customers with due dates in $[t, t + r/h]$ can be served with holding cost at most $2r$ per unit. By definition of X^i, this means that the customers in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$ will be served with total holding costs at most $2r$ per unit. The remaining customers left to be served are

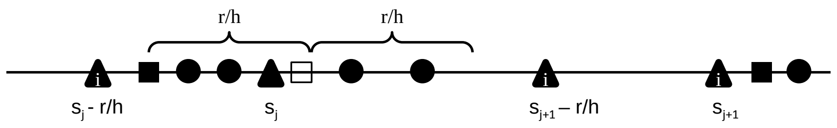

Figure 2-3: In this figure, the triangles are orders, the circles are due dates for customers in $\mathcal{A}^i$, the solid squares are times in T^i, and the hollow squares are times in $\hat{T}^i$ that were pruned. In this picture, only the orders with an i written contain an order of type i. The first pair of orders is associated with Case 2 and the second pair of orders is associated with Case 1. The first two demands are in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$. The next two demands are in $k \in Y^i \cap \mathcal{R}^*$. The last demand is in $\mathcal{A}^*$.

those in $Y^i \cap \mathcal{R}^*$. Consider customer $k \in Y^i \cap \mathcal{R}^*$. By construction of T^i, it follows that there exists a $t \in T^i$ such that $t \leq a_k \leq t + r/h \leq a_k + r/h$. In addition, from the definition of a_k and Lemma 9, it follows that $t_k \in [a_k, a_k + r/h]$. By the property assumed above, there exists an order that includes item type i within r/h before t. The holding cost from that order to t, t to a_k, and a_k to t_k are each at most r per unit. Thus, customer k can be served with a holding cost of at most $3r$ per unit. (See Figure 2-3 for an example.)

It is now sufficient to ensure that indeed for each $t \in T^i$, there exists an order of type i within r/h time periods earlier than t. To achieve this, extra item orders will be added to the construction. From Lemma 10, it follows that the setup interval corresponding to t intersected some optimal setup interval starting at s_j. (If there is a choice of intersections, choose s_j that contains an item order of type i if one exists.) From Lemma 9, it follows that $s_j - r/h \leq t \leq s_j + r/h$. We now consider two cases and show how to enforce the property in each case.

Case 1: There is a type i customer in $\mathcal{A}^*$ with due date in $[t, t+r/h]$. By construction, this implies that there are type i orders at s_j and $s_j - r/h$, respectively.

Case 2: There is no type i customer in $\mathcal{A}^*$ with due date in $[t, t + r/h]$**.** If $s_j - r/h \leq t < s_j$, then we place an *extra* item order of type i at the joint order located at time $s_j - r/h$. Otherwise, $s_j \leq t \leq s_j + r/h$ and we place the extra order of type i at s_j. Since t corresponds to a setup interval from a stable pair solution, it follows from (2.1) that there exists a set of customers with due dates in $[t, t + r/h]$ whose rejection costs are greater than K_i. Under the case assumption, the type i

demands with due dates in $[t, t + r/h]$ are all in $\mathcal{R}^*$. Furthermore, all customers with due dates in $[t, t + r/h]$ are in X^i. Thus, the extra item orders have cost at most $R(X^i \cap \mathcal{R}^*)$. Each customer in $X^i \cap \mathcal{R}^*$ can be used at most once to pay for an extra item order by the spacing of the times we enforced in the construction of T^i.

The total cost incurred by the construction is $K^* + H^* + K^* + \sum_{i=1}^{M}(R(X^i \cap \mathcal{R}^*) + 2R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*) + 3R(Y^i \cap \mathcal{R}^*)) \leq 2P(\mathcal{A}^*) + R(\mathcal{R}^*) + 2R(\mathcal{A} \cap \mathcal{R}^*)$ which completes the proof. $\square$

Combining Theorems 1 and 2 with Lemma 12 obtains the following main result.

Theorem 7. *The Copycat and StablePair Algorithms for the Joint Replenishment problem with Online Customer Selection are 4-competitive and 3-competitive, respectively.*

Note that the analysis is tight for the StablePair Algorithm by Example 8 in Appendix A.1 since JR is a generalization of ELS. Theorem 8 below, proved in Appendix A.2, gives a slightly weaker result for the case when each item type i has a specific per unit rejection cost r^i and a specific per unit holding cost h^i.

Theorem 8. *The Copycat and StablePair Algorithms for the JR problem with Online Customer Selection and item-specific holding and rejection costs are ($3+ \frac{max_i r^i/h^i}{min_i r^i/h^i}$)-competitive and ($2+ \frac{max_i r^i/h^i}{min_i r^i/h^i}$)-competitive, respectively.*

2.7.5 FL Analysis with Fixed Rejection Costs

Now we will bound the production costs incurred by the StablePair Algorithm for the FL problem with online customer selection. This will require the two following lemmas which are proved in Appendix A.2 and are similar in spirit to Lemmas 9 and 10.

Lemma 13. *Let (Q, T) be a stable pair. For each customer $k \in T$, the per unit service cost incurred by the FL solution to $P(Q, T)$ is at most r.*

For each customer $k \in \mathcal{A}$ accepted by the StablePair Algorithm, define (Q_k, T_k) to be the stable pair used by StablePair to accept k. Let $S_k \subseteq T_k$ be the set of customers served by the same facility as k in $P(Q_k, T_k)$ and let q_k denote this shared facility.

Lemma 14. *For each $k \in \mathcal{A}$, the set $S_k \cap \mathcal{A}^*$ is nonempty.*

We now proceed to bound the production costs of StablePair and therefore Copycat.

Lemma 15. *If the StablePair (Copycat) Algorithm is used for the FL problem with online customer selection, then $P(\mathcal{A}) \leq P(\mathcal{A}^*) + 3R(\mathcal{A} \cap \mathcal{R}^*) \leq 3C^*(U)$.*

Proof. We will construct a feasible solution to serve the customers in $\mathcal{A}$ with cost at most three times the optimal offline cost $C^*(U)$. We simply open all the facilities in the FL solution to $P(\mathcal{A}^*)$ and serve all the customers in $\mathcal{A}$ by the nearest facility.

By construction, the facility costs plus the service costs for $\mathcal{A} \cap \mathcal{A}^*$ is bounded by $P(\mathcal{A}^*)$. Now consider a customer $k \in \mathcal{A} \cap \mathcal{R}^*$. From Lemma 14, it follows that there exists a customer $l \in S_k \cap \mathcal{A}^*$. Let q_l^* denote the facility from which customer l is served in $P(\mathcal{A}^*)$. Lemma 13 implies that $c(k, q_k)$, $c(q_k, l)$, and $c(l, q_l^*)$ are all at most r. Since c is a metric, it follows that $c(k, q_l^*) \leq c(k, q_k) + c(q_k, l) + c(l, q_l^*) \leq 3r$. Thus, the per unit service cost for customer k is at most $3r$. (See Figure 2-4 for an example.) This completes the proof since the construction has cost at most $P(\mathcal{A}) \leq P(\mathcal{A}^*) + 3R(\mathcal{A} \cap \mathcal{R}^*) \leq 3C^*(U)$. $\square$

Combining Theorems 1 and 2 with Lemma 15 obtains the following main result.

Theorem 9. *The Copycat and StablePair Algorithms for the Facility Location problem with Online Customer Selection are 4-competitive and 3-competitive, respectively.*

2.7.6 Improving FL Analysis via Scaling

The results for FL with online customer selection can be improved via a scaling technique to get a competitive ratio of $1 + \sqrt{2}$. In particular, the rejection costs are scaled by a factor $\alpha > 0$, and then the StablePair Algorithm is applied to the scaled

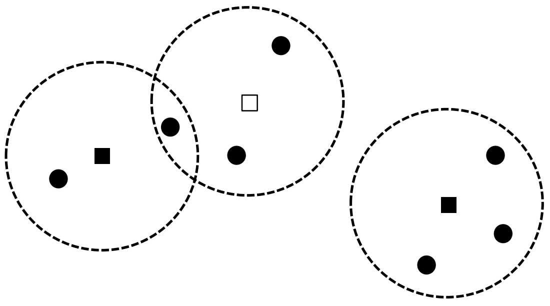

Figure 2-4: In this figure, the squares represent facilities and the circles represent locations for customers in $\mathcal{A}$. The solid squares represent open facilities and the hollow squares are closed facilities. The circle around each facility represents the service area where the service cost per unit is at most r from the respective facility. The customers in the circles corresponding to open facilities are in $\mathcal{A} \cap \mathcal{A}^*$. The remaining customers in the circles corresponding to closed facilities are in $\mathcal{A} \cap \mathcal{R}^*$ and are no more than $3r$ away from an open facility.

input. The purpose of this idea is to "hedge" against the future by giving a different weighting to the rejection costs. Drop the superscript S and denote $\mathcal{A}_\alpha$ and $\mathcal{R}_\alpha$ as StablePair's decisions under the scaled input. In Lemma 16 below, we obtain a new bound for the rejection costs. The proof is in Appendix A.2.

Lemma 16. *The StablePair Algorithm with the rejection costs scaled by a parameter α has total rejection cost $R(\mathcal{R}_\alpha) \leq R(\mathcal{R}_\alpha \cap \mathcal{R}^*) + \frac{1}{\alpha} P(\mathcal{R}_\alpha \cap \mathcal{A}^*)$.*

Note that the bound on production costs in Lemma 15 is unbalanced. By scaling down the rejection costs by a factor $\alpha < 1$, we can obtain the following lemma which we prove in Appendix A.2.

Lemma 17. *The StablePair Algorithm with scaling for the FL problem with online customer selection has production costs $P(\mathcal{A}_\alpha) \leq P(\mathcal{A}^*) + (2\alpha + 1) R(\mathcal{A}_\alpha \cap \mathcal{R}^*)$ when $\alpha < 1$.*

Combining Lemmas 16 and 17 obtains the following theorem, whose analysis is tight according Example 10 in Appendix A.1.

Theorem 10. *The StablePair Algorithm with scaling for the Facility Location problem with online customer selection is $(1 + \sqrt{2})$-competitive when $\alpha = \frac{1}{\sqrt{2}}$.*

2.7.7 Computational Results

In this section, we present the results of computational experiments that test the typical empirical performance of our algorithms on the Economic Lot Sizing Problem with Online Customer Selection. We carried out three different scenarios of the same experiment. The parameters for each scenario were $K = 100, h = 1$, and $r = 5$. The time horizon was for one month with periods from 1 to 30. There were $N = 500$ customer arrivals simulated for each scenario. Copycat, StablePair, and StablePair with scaling factor of two (the rejection cost of each customer is weighted double the true value) were each tested for the same sequence of customer arrivals. After every customer k, the performance ratio, $C(U_k)/C^*(U_k)$, was calculated. Note that we are comparing ourselves to the optimal offline solution, which is a benchmark that is impossible to achieve. For all three scenarios, the due dates of the customers were drawn uniformly at random across the time horizon. We believe this distribution is the worst case for the supplier since she can never make a meaningful estimate of where the next customer's due date will be. In the first scenario, 'Conservative', the demands of each customer comprise of one unit ($d_k = 1$). In the second scenario, 'More Demands', the demand quantity is drawn uniformly at random from 1 to 10. Finally, in the third scenario, 'Large Orders First', the customers also have a random demand quantity between 1 and 10 but the first three orders are very large orders placed every two weeks, i.e. at times 1, 15, and 29, representing the realistic situation where the supplier has at least one major routine customer. For each scenario, the results were averaged over 100 different customer sequences. The results are in Figure 2-5.

Based on these experiments, we see that even in the 'Conservative' scenario, the costs of Copycat and StablePair are at most 1.5 times larger than the optimal offline cost (which is not achievable in practice). The upper bound of 1.5 held up in all 25 runs of the scenario. Note that the performance ratios are much lower than the theoretical guarantee of 3. The 'Conservative' graph in Figure 2-5 indicates that Copycat and StablePair start out perfect since both OPT and the algorithms are rejecting

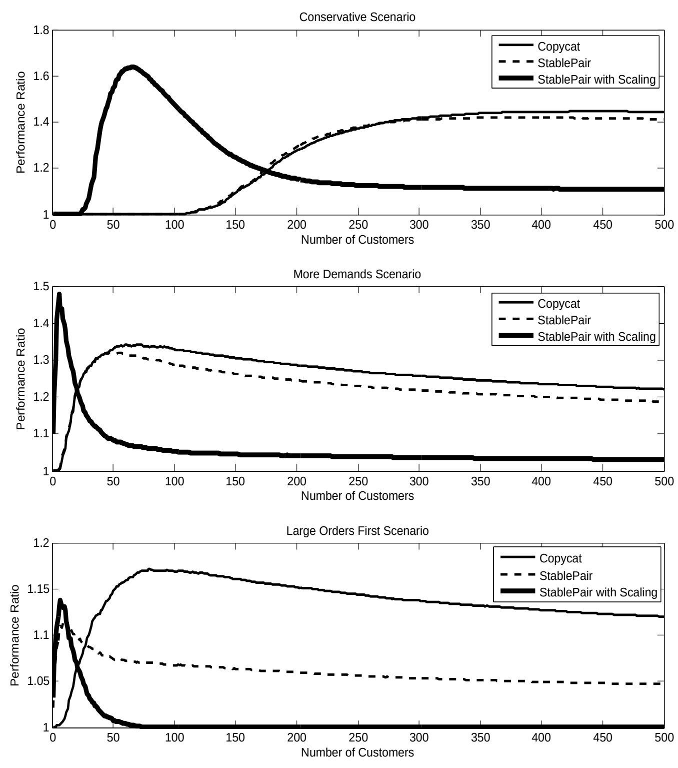

Figure 2-5: Three experiments detailing the actual performance ratio of three algorithms on the ELS problem with online customer selection.

everything because there are not enough customers to warrant production. Then around 125 customer arrivals OPT begins accepting some customers while Copycat and StablePair do the same and therefore begin incurring lots of fixed costs on top of the rejection costs. Eventually, as the number of customers gets very large, OPT, Copycat, and StablePair begin to accept everything and the performance ratios stabilize. (Note that $C^*(U_k)$ converges to TK while the costs of the algorithms are TK plus some initial rejection costs.) Since StablePair accepts a superset of what Copycat accepts, then StablePair outperforms Copycat as the number of customers gets large. StablePair with scaling does poorly in the beginning since it accepts customers before OPT does, but later on it significantly outperforms the other algorithms.

Similar observations hold for the 'More Demands' and 'Large Orders First' graphs in Figures 2-5, except that the benefits of scaling are not as significant since more customers were accepted early on which makes it more difficult to make errors. Specifically, Copycat/StablePair accept more customers when they come in larger batches, so as the number of customers gets large this improves the performance ratio significantly. If there is some information known about the customer process, this can help decide on a reasonable choice for the scaling factor. The more customers one anticipates, the larger the scaling factor should be. Also note that in addition to StablePair being computationally much faster than Copycat, it typically outperforms Copycat. In Appendix A.3, we provide tables describing the performance of our algorithms for the same scenarios but with different values of r.

2.8 FairShare Algorithm

The FairShare Algorithm is based on the repeated simulation of specific cost sharing mechanisms. Suppose we are given a set of players U and a cost function $P(\cdot)$. A *cost-sharing mechanism* is a mechanism that first collects bids from each player in U, and then decides which subset of players to serve. In addition, the mechanism assigns a cost share (price) to each serviced player. If the mechanism chooses to serve $T \subseteq U$, then we let $\chi(i, T)$ denote the cost share assigned to player $i \in T$ for service. We refer to the function χ as a cost sharing scheme (or method), which can possibly take on some of the properties discussed later on. In the next section we describe several useful cost sharing methods for various production cost functions of interest.

Given a set of customers T, a cost-sharing scheme χ, and a scalar c, we let $M(\chi, T, c)$ denote the set of customers output by the following procedure, known as a *Moulin mechanism* (Moulin [63]). A Moulin mechanism is a class of cost sharing mechanisms where every player begins by receiving service, and players are iteratively removed if their cost share is higher than their (scaled) bid. The mechanism terminates when every remaining player has a cost share less than or equal to his (scaled) bid, which we denote by r_i for player i for convenience.

Moulin Mechanism: $M(\chi, T, c)$

```
1. Collect bids $r_i$ from each player $i \in T$.
2. If $cr_i \geq \chi(i, T)$ for all $i \in T$, halt and output $T$.
3. Else, let $j \in T$ be a player with $cr_j < \chi(j, T)$. Set $T := T \backslash \{j\}$.
4. Go to Step 2.
```

Now we can define our general online customer selection algorithm, which we call FairShare, according to the following procedure. FairShare relies on a cost sharing method χ and a scalar c, where c is a cost scaling parameter chosen to optimize performance. The cost sharing method χ must be specific for the production cost function $P(\cdot)$ being considered in the application, and will act as a proxy for how much each customer is contributing to the overall production cost of the accepted customers. For example, we will use the facility location cost sharing method of Pál and Tardos [68] for running the FairShare on the facility location problem with online customer selection. For the current customer k, the FairShare algorithm is simulating a Moulin mechanism with χ as the cost sharing method, the entire set of observed customers so far U_k as the players, and their respective rejection costs as their bids. If the simulated mechanism accepts customer (player) k, then so does FairShare and vis versa.

FairShare Accept current customer k if and only if $k \in M(\chi, U_k, c)$.

Clearly the FairShare algorithm converges in at most N iterations since there is at least one person removed in every iteration except for the very last one. Furthermore, since all of the cost sharing schemes we consider can be computed efficiently in polynomial time, then FairShare is generally an efficient algorithm. We now define some desirable properties for χ. A cost sharing scheme χ is

1. *cross-monotonic* if $\chi(i,S) \geq \chi(i,T)$ for all $i \in S \subseteq T \subseteq U$.

2. (β,γ)-*budget balanced* if $\frac{P(T)}{\gamma} \leq \sum_{i\in T}\chi(i,T) \leq \beta P(T)$

3. α-*summable* for a monotonically increasing function $\alpha(\cdot)$ if $\sum_{l=1}^{|T|}\chi(i_l,T_l) \leq \alpha(|T|)P(T)$ for every ordering σ of U and every set $T \subseteq U$, where T_l denotes the set of the first l demands in T and i_l denotes the lth demand of T (with respect to σ).

The cross-monotonicity property simply states that if the number of players receiving service grows, then the cost share of each individual will not increase. In other words, if we remove a player, then the cost share of all remaining players can only increase. In the mechanism design setting, if χ is cross-monotonic, then the Moulin mechanism will induce truthful bidding (Moulin and Shenker [64]).

The (β,γ)- budget balanced property ensures that the sum of the cost shares is no more than β times the associated cost, but also recovers at least $1/\gamma$ of the cost. A $(1,\gamma)$-budget balanced is said to be *competitive* while a $(\beta,1)$-budget balanced mechanism is said to be *no-deficit.* Although χ is providing a specific approximation to $P(\cdot)$, we use the exact value of $P(\cdot)$ computing the values of $C^*(\cdot)$ and $C(\cdot)$.

Finally, the α-summable property is a measure of the efficiency (social welfare), of the Moulin mechanism, as described by Roughgarden and Sundararajan [73]. To see this, pick an arbitrary order σ of $T \subseteq U$, and let each player l's valuation, and therefore bid, be $\chi(l,T_l) - \epsilon$ for some arbitrarily small ϵ. If we run the Moulin mechanism $M(\chi,T,1)$, then no player will be chosen, since they will be eliminated in reverse order by construction. Therefore the total cost of the system will be roughly $\sum_{l=1}^{|T|}\chi(i_l,T_l)$, where the optimal cost would be just $C(U)$. If σ is chosen adversarily, then $\alpha(\cdot)$ represents the worst possible situation that we can construct using the previous example. (A more rigorous analysis is in Roughgarden and Sundararajan [73].) We note that contrary to β and γ, α is typically not a scalar but rather a function that depends on the number of customers being served.

Given a cost sharing scheme χ that satisfies these three properties, we can intuitively explain why we can expect the FairShare algorithm to perform well. First, the cross-monotonicity implies that once a player is accepted by FairShare, he will always be accepted in every simulated output of the Moulin mechanism after his arrival. Thus, the current set of accepted customers is always a subset of the current output of the Moulin mechanism, whose costs can be bounded by the other two properties. In essence, cross-monotonicity implies that χ is a *fair* way to *share* the costs among the customers, since as more customers are added, the less each customer is 'charged' for service. Due to the (β, γ)-budget balanced property, we can expect that the production costs for serving the accepted customers will be not too large. Finally, motivated by the previous example, the α-summability directly characterizes how much we will expect to incur in rejection costs. In the next section, we formally provide theoretical guarantees on the cost of the FairShare algorithm. We explicitly characterize the competitive ratio of FairShare as a function of α, β, γ, c, and N. In the next section, we discuss specific cost sharing methods from the literature that yield relatively small competitive ratios.

2.8.1 Analysis

In this section, we will provide explicit bounds on both the production and rejection costs incurred by FairShare. We assume that we have a (β, γ)-budget balanced, α-summable, cross-monotonic, cost sharing scheme χ for the production cost function $P(\cdot)$. Remember that U_k refers to the first k customers that have arrived. We will let $\mathcal{A}_k^M = M(\chi, U_k, c)$ and $\mathcal{R}_k^M = U_k \backslash M(\chi, U_k, c)$ for a given choice of parameter c to be optimized later. A set with no subscript simply refers to the final customer N. We first prove a key lemma which says that the the sets $\mathcal{A}_k^M$ are monotonically increasing in k. Intuitively, the reason for this is that as more customers are added, the better off everyone is due to the cross-monotonicity of χ. This will be key to prove our production bound in Lemma 19.

Lemma 18. *The accepted set of customers by $M(\chi, U_k, c)$ is monotonically increasing* *in* k*, i.e.,* $\mathcal{A}_1^M \subseteq A_2^M \subseteq \ldots \subseteq A_N^M$.

Proof. We will prove the lemma by induction. The base case is trivial since we start the problem with no customers. Assume $\mathcal{A}_1^M \subseteq A_2^M \subseteq \ldots \subseteq A_k^M$ and we want to show that $\mathcal{A}_k^M \subseteq \mathcal{A}_{k+1}^M$. Assume for contradiction that a customer $j \in \mathcal{A}_k^M$ was not in $\mathcal{A}_{k+1}^M$. In case there are multiple choices for j, we choose the j that was first removed by the Moulin mechanism for $M(\chi, U_{k+1}, c)$. Let S_j denote the set of players that had not been rejected by the Moulin mechanism for $M(\chi, U_{k+1}, c)$ just before j was removed. By construction, $j \in S_j$. Then we know that

$$cr_j < \chi(j, S_j) \leq \chi(j, \mathcal{A}_k^M) \leq cr_j.$$

The first inequality follows from the fact that j was removed by the mechanism $M(\chi, U_{k+1}, c)$ when the current remaining set at the time was S_j. The second inequality follows from the cross-monotonicity of χ and the fact that by definition $\mathcal{A}_k^M \subseteq S_j$ by construction. The last inequality follows from the fact that the set $\mathcal{A}_k^M$ was previously output by the Moulin mechanism. Thus, we arrived at a contradiction, which implies that each $j \in \mathcal{A}_k^M$ is also in $\mathcal{A}_{k+1}^M$, and thus $\mathcal{A}_k^M \subseteq \mathcal{A}_{k+1}^M$. □

Using this lemma, we can now prove a bound on the production costs of FairShare.

Lemma 19. *The production costs of the FairShare algorithm are* $P(\mathcal{A}) \leq \gamma\left(cR(\mathcal{R}^*) + \beta P(\mathcal{A}^*)\right)$.

Proof. The production cost of serving the accepted customers is

$$\begin{aligned}
P(\mathcal{A}) &\leq P(\mathcal{A}^M) \\
&\leq \gamma \sum_{k \in \mathcal{A}^M} \chi(k, \mathcal{A}^M) \\
&\leq \gamma \left(\sum_{k \in \mathcal{A}^M \cap \mathcal{R}^*} \chi(k, \mathcal{A}^M) + \sum_{k \in \mathcal{A}^M \cap \mathcal{A}^*} \chi(k, \mathcal{A}^M \cap \mathcal{A}^*) \right) \\
&\leq \gamma \left(\sum_{k \in \mathcal{A}^M \cap \mathcal{R}^*} \chi(k, \mathcal{A}^M) + \beta P(\mathcal{A}^M \cap \mathcal{A}^*) \right) \\
&\leq \gamma \left(\sum_{k \in \mathcal{A}^M \cap \mathcal{R}^*} \chi(k, \mathcal{A}^M_k) + \beta P(\mathcal{A}^M \cap \mathcal{A}^*) \right) \\
&\leq \gamma \left(\sum_{k \in \mathcal{A}^M \cap \mathcal{R}^*} cr_k + \beta P(\mathcal{A}^M \cap \mathcal{A}^*) \right) \\
&= \gamma \left(cR(\mathcal{A}^M \cap \mathcal{R}^*) + \beta P(\mathcal{A}^M \cap \mathcal{A}^*) \right) \\
&\leq \gamma \left(cR(\mathcal{R}^*) + \beta P(\mathcal{A}^*) \right).
\end{aligned}$$

The first inequality follows since Lemma 18 implies $\mathcal{A} \subseteq \mathcal{A}^M$ and $P(\cdot)$ is monotonic. The second inequality follows from χ being (β, γ)-budget balanced. The third inequality follows from cross-monotonicity of χ. The fourth inequality follows from χ being (β, γ)-budget balanced. The fifth inequality follows Lemma 18 which implies $\mathcal{A}^M_k \subseteq \mathcal{A}^M$ and the cross-monotonicity of χ. The sixth inequality follows from the fact that $\mathcal{A}^M_k$ is an output of the Moulin mechanism. The equality follows from the definition of $R(\cdot)$. The last inequality follows from the monotonicity of $R(\cdot)$ and $P(\cdot)$. □

We now show that any subset of the rejected customers of FairShare will be rejected by the Moulin mechanism as well. Intuitively this is also due to the cross-monotonicity of χ since these customers were already being rejected when they were in consideration with an even larger set of customers. This result will be useful in proving our rejection cost bound in Lemma 21.

Lemma 20. *For any subset of customers $Q \subseteq \mathcal{R}$, the Moulin mechanism for*

$M(\chi, Q, c)$ outputs the empty set.

Proof. Assume for contradiction that the Moulin mechanism for $M(\chi, Q, c)$ outputs $T \neq \emptyset$. Let k be the last customer that arrived in T. Consider the first time a customer $j \in T$ was removed by $M(\chi, U_k, c)$ and let T_j denote the players remaining just before j was removed. Note since $k \in \mathcal{R}$ that j and T_j are well-defined. Then we know that

$$cr_j < \chi(j, T_j) \leq \chi(j, T) \leq cr_j.$$

The first inequality from the fact j was removed by the Moulin mechanism for $M(\chi, U_k, c)$ when the current remaining set at the time was T_j. The second inequality follows from the fact that $T \subseteq T_j$ by construction and the cross-monotonicity of χ. The last inequality follows from the fact that T is an output of the Moulin mechanism. Thus we arrived at a contradiction and conclude that T must be indeed empty. □

Using the previous lemma, we can now obtain a bound the rejection costs of the FairShare algorithm.

Lemma 21. *The rejection costs of the FairShare algorithm are $R(\mathcal{R}) \leq R(\mathcal{R}^*) + \frac{\alpha(N)}{c} P(\mathcal{A}^*)$.*

Proof. Let $Q = \mathcal{R} \cap \mathcal{A}^*$ which is the set of customers that FairShare rejected but the optimal offline solution accepted. From Lemma 20, we know that $M(\chi, Q, c)$ returns the empty set. For each $j \in Q$, let Q_j be the remaining set of customers just before customer j was rejected in the Moulin mechanism for $M(\chi, Q, c)$. Then by construction we know that

$$cr_j < \chi(j, Q_j).$$

Then we can show that

$$\begin{aligned} R(\mathcal{R} \cap \mathcal{A}^*) &= R(Q) \\ &< \sum_{j \in Q} \frac{\chi(j, Q_j)}{c} \\ &\leq \frac{\alpha(|Q|)}{c} P(Q) \\ &\leq \frac{\alpha(N)}{c} P(\mathcal{A}^*). \end{aligned}$$

The first equality follows from the definition of Q. The first inequality follows from the previous inequality we derived. The second inequality follows from α-summability of χ. The last inequality follows from the monotonicity of $\alpha(\cdot)$ and $P(\cdot)$. Adding $R(\mathcal{R} \cap \mathcal{R}^*) \leq R(\mathcal{R}^*)$ to both side of the inequality completes the proof. □

Using the bounds we have obtained for the FairShare algorithm, we can now easily prove the general theorem below.

Theorem 11. *Given a (β, γ)-budget balanced, α-summable, cross-monotonic cost sharing scheme χ for a production problem P with online customer selection, the FairShare algorithm is* $\max(1 + \gamma c, \frac{\alpha(N)}{c} + \gamma\beta)$*-competitive.*

Proof. We have that

$$\begin{aligned} C(U) &= R(\mathcal{R}) + P(\mathcal{A}) \\ &\leq R(\mathcal{R}^*) + \frac{\alpha(N)}{c} P(\mathcal{A}^*) + P(\mathcal{A}) \\ &\leq R(\mathcal{R}^*) + \frac{\alpha(N)}{c} P(\mathcal{A}^*) + \gamma\left(cR(\mathcal{R}^*) + \beta P(\mathcal{A}^*)\right) \\ &\leq \max(1 + \gamma c, \frac{\alpha(N)}{c} + \gamma\beta)(R(\mathcal{R}^*) + P(\mathcal{A}^*)) \\ &= \max(1 + \gamma c, \frac{\alpha(N)}{c} + \gamma\beta) C^*(U). \end{aligned}$$

The first equality follows from the definition of $C(\cdot)$. The first inequality follows from Lemma 21. The second inequality follows from Lemma 19. The third inequality is combining terms. The last equality follows from the definition of $C^*(\cdot)$ which completes the proof. $\square$

The following corollary states what happens if we want to use an approximation algorithm or online algorithm for P.

Corollary 1. *If a d-approximation or d-competitive algorithm is used to compute $P(\cdot)$, then the FairShare algorithm is* $\max(1 + \gamma c d, \frac{\alpha(N)}{c} + \gamma \beta d)$*- competitive.*

Finally, if the number of customers is known up to a constant factor, then c can be chosen to be $\Theta(\sqrt{\alpha(N)})$ which optimizes the bound in Theorem 11. In particular, knowing N approximately allows for a substantial improvement in the competitive ratio from $O(\alpha(N))$ to $O(\sqrt{\alpha(N)})$.

Corollary 2. *If N is known up to constant factors, and β and γ are scalars, then choosing $c = \Theta(\sqrt{\alpha(N)})$ makes the FairShare algorithm $O(\sqrt{\alpha(N)})$-competitive.*

We now provide an example on how to apply these results. (See Table 2 in Section 4 for more results.) Consider the FL problem with online customer selection. [68] provided a cross-monotonic, (3,1)-budget balanced cost sharing scheme for the facility location problem. [74] shows that the summability factor α(N) $= \log(N)$. Setting $c = \sqrt{\log N}$ and using Corollary 2 gives an overall competitive ratio of $O(\sqrt{\log N})$.

The table below summarizes the results for some problems with arbitrary rejection costs. We will make the assumption that each customer requests only one unit, and discuss how to relax this assumption in the next section. This assumption is due to the fact that the cost sharing schemes we use also make this single unit per customer assumption. The first column denotes the production cost problem along with the citation of the cost sharing method that we use in FairShare. The second column displays the summability function α for the corresponding problem along with the paper that showed this. The third and fourth columns show the β and γ factors that were shown in the corresponding cost-sharing method paper. The fifth column denotes the optimal choice for c. The sixth column denotes the competitive ratio achieved by using FairShare with the corresponding cost sharing method and choice of c (we assume we know $\Theta(N)$ and use Corrollary 2). Note that the results for facility location and economic lot sizing assuming each customer requests one unit, we discuss extensions in Section 2.8.2. The final column shows lower bounds on the best possible competitive ratios, shown in Section 2.9.2 and Appendix B.

Problem	α	β	γ	c	FairShare	Lower Bound
FL [68]	$\log N$ [74]	3	1	$\sqrt{\log N}$	$O(\sqrt{\log N})$	$\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$
ELS [91]	$\log N$ [74]	$\max(\frac{b}{h}, \frac{h}{b})$	1	$\sqrt{\log N}$	$O(\sqrt{\log N})$	$\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$
ST [48]	$\Theta(\log^2 N)$ [73]	2	1	$\log N$	$O(\log N)$	$\Omega(\sqrt{\log N})$
SF [56]	$\Theta(\log^2 N)$ [21]	2	1	$\log N$	$O(\log N)$	$\Omega(\sqrt{\log N})$
SSROB [42]	$\Theta(\log^2 N)$ [74]	4.6	1	$\log N$	$O(\log N)$	$\Omega(\sqrt{\log N})$
MROB [74]	$\Theta(\log^2 N)$ [74]	$O(1)$	1	$\log N$	$O(\log N)$	$\Omega(\sqrt{\log N})$

Based on the results of Brenner and Schäfer [13] and Bleischwitz and Schoppmann [12], we can also get $O(\sqrt{\log N})$ competitive ratios with FairShare for the machines scheduling problems with online customer selection using identical and related machines, respectively. Finally, it is worth noting that using the cost sharing method of Moulin and Shenker [64] we can get $O(\sqrt{\log N})$ competitive ratios for submodular problems with online customer selection, although this does not match the optimal guarantee of 2 in Elmachtoub and Levi [27]. To the best of our knowledge, the results in the last two columns are the first of their kind with one exception in that Qian and Williamson [71] found a similar guarantee for the Steiner tree problem with online customer selection.

2.8.2 Extensions for Online Market Selection

In this section, we consider a more typical scenario where each customer actually has a multi-unit request that can even be across multiple time periods or locations. We refer to this type of customer as a market, which requires special consideration since cost sharing schemes have typically only been developed with single demand customers in mind. Geunes et al. [38] gives a 2.06-approximation algorithm for several problems with *offline* market selection. In order to handle *online market selection*, we simply create a new cost sharing scheme that is based on summing up the cost shares for each demand when the original cost sharing scheme is used.

For example, assume we have a cross-monotonic, (β, γ)-budget balanced, α-summable cost sharing scheme $\hat{\chi}$. However, $\hat{\chi}$ assumes that each player has one demand request. If customers have multiple demand requests, which create markets, and they only want to be served if their entire market of demands can be served, then we need to come up with a new cost sharing scheme. Specifically, for each customer j, let T_j be the set of demands that customer j wants. We will define the new cost sharing scheme χ as $\chi(j, \cdot) = \sum_{k \in T_j} \hat{\chi}(k, \cdot)$. Clearly χ maintains the cross-monotonicity and (β, γ)-budget balanced properties. It remains α summable, but the input to $\alpha(\cdot)$ is now the total number of demands, rather than the number of customers. Thus, the competitive ratios achieved by FairShare are now dependent on the total number of units of demand requested. In addition, our lower bounds also have the feature that the lower bound will also depend on the the total number of units of demand request. Specifically, all our results are trivially extended by just replacing N by the total number of demands requested.

2.9 Lower Bounds

2.9.1 Submodular, ELS, JR, and FL with Fixed Rejection Costs

In this section we show that no deterministic algorithm has a competitive ratio better than two for any of the online customer selection problems that we have considered under the Copycat and StablePair Algorithms.

Theorem 12. *The competitive ratio for any deterministic algorithm used to solve submodular, ELS, FL, or JR problems with online customer selection is at least 2.*

Proof. Consider the To Build or Not to Build problem with online customer selection, as explained in Example 1. Let $K = 1$ and assume that the rejection cost for each customers k is $r_k = \epsilon$ for some $\epsilon > 0$. It can be seen that this is a special case of a submodular, ELS, JR, or FL problem with online customer selection. Any nontrivial deterministic algorithm for this problem simply specifies how many arrivals have to occur before it starts accepting customers. We denote ALG_q as the algorithm which rejects the first $q-1$ customers and then accepts all remaining customers starting with customer q. Thus, all algorithms can be parameterized by $q \in \mathbb{Z}_+ \cup \{\infty\}$. Now focus on the worst-case adversary strategy given a fixed q.

Case 1 If $q > 2/\epsilon$, then the adversary generates $2/\epsilon$ arrivals. Thus ALG_q will reject all q customers and incur a total cost of 2, while the optimal offline cost is clearly 1.

Case 2 If $1/\epsilon < q \leq 2/\epsilon$, then the adversary generates exactly q arrivals. Thus ALG_q incurs a cost of $1+(q-1)\epsilon$, while the optimal offline cost is just 1.

Case 3 If $q \leq 1/\epsilon$, then the adversary generates exactly q arrivals. Thus ALG_q incurs a cost of $1+(q-1)\epsilon$, while the optimal offline cost is $q\epsilon \leq 1$.

The competitive ratio in all three cases is at least 2 or arbitrarily close to 2 as ϵ approaches 0. □

2.9.2 FL and ELS with Arbitrary Rejection Costs

In this section, we provide lower bounds on the competitive ratio for any algorithm, deterministic or randomized, for the facility location problems with online customer selection. The proof for the economic lot sizing problem is almost identical and therefore omitted. In Appendix B, we provide the proof for the Steiner tree problem which has many similar features to the proof below. The technique relies on constructing an instance similar to the one used in the lower bound proof for the online facility location problem without customer selection in Fotakis [34]. However, we modify the demand instance and add rejection costs in order to obtain a worst case adversary.

Theorem 13. *The competitive ratio for any deterministic or randomized algorithm for the facility location problem with online customer selection is* $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$.

Proof. To show this lower bound, we need to specify an instance of the problem such that no algorithm can come with in a factor of $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$ of the optimal offline cost, where N is the total number of customers. The proof is by construction, and we start by building a hierarchically well-separated binary tree (Bartal [5]) of depth h (Figure 2-6), where h will be specified later. The root node will be level 0, its children will be level 1, and so on. There is a potential facility at each leaf in the tree, each with a facility cost f. The distance from a level-i node to its children will be D/m^i, where D and m will also be specified later. For a node v in the tree, we define T_v to be the subtree rooted at v. We now state two easily verifiable facts about the tree.

1. Let v be a level-i node. Then the distance from v to a node not in T_v is at least D/m^{i-1}.

2. Let v be a level-i node. Then the distance from v to any node in T_v is at most $D/(m^{i-1}(m-1))$.

We will now provide a distribution of customer sequences and give a lower bound of $\Omega\left(\sqrt{\frac{\log N}{\log\log N}}\right)$ on the competitive ratio of any deterministic algorithm. Using Yao's principle (Yao [94]), this achieves the desired result. The distribution can be described according to the following procedure. See Figure 2-6 for an example. The sequence $v_1, \ldots, v_h$ denotes the locations of where the customers' locations will occur, and v is simply keeping track of how to move along the tree. We will assign each v_i to either $V^{\hat{\mathcal{A}}}$ or $V^{\hat{\mathcal{R}}}$ depending on whether we will accept or reject it in the feasible offline solution we will consider.

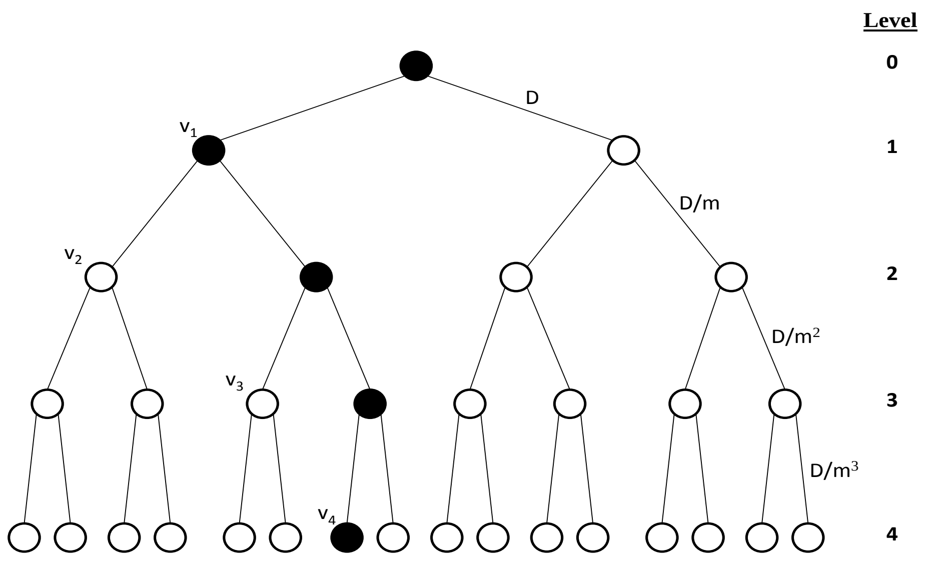

Figure 2-6: In this figure, each circle represents a node, and the lines represent where the node can travel to. The black nodes denote a random path that the v node took in the demand sequence construction. In this example, $V^{\hat{A}} = \{v_1, v_4\}$ and $V^{\hat{R}} = \{v_2, v_3\}$.

1. Let v :=root, $i := 0$, $V^{\hat{\mathcal{A}}} = \emptyset$ and $V^{\hat{\mathcal{R}}} = \emptyset$.
2. Let $i = i+1$. Set v_i :=left child of v. Generate m^{i-1} consecutive demands at v_i each with rejection cost of $r_i = D/(m^{i-1}\sqrt{m})$.
3. With probability $1-1/\sqrt{m}$, set v :=left child of v and let $V^{\hat{\mathcal{A}}} = \{v_i\} \cup V^{\hat{\mathcal{A}}}$. Otherwise, with probability $1/\sqrt{m}$, set v :=right child of v and $V^{\hat{\mathcal{R}}} = \{v_i\} \cup V^{\hat{\mathcal{R}}}$.
4. If $i < h$, go to Step 2. Else, Stop.

We now compute an upper bound on the expected cost of the optimal offline solution, $\mathbb{E}[\text{OPT}]$, by constructing a feasible solution for any sample path. Let $\hat{\mathcal{A}}$ denote all the customers located at a node in $V^{\hat{\mathcal{A}}}$ and let $\hat{\mathcal{R}}$ denote all the customers located at a node in $V^{\hat{\mathcal{R}}}$. In our feasible solution, we will accept $\hat{\mathcal{A}}$ and reject $\hat{\mathcal{R}}$. Note that there is a common facility in the subtrees of all the nodes that have customers in $\hat{A}$, which is where how they will be served in this feasible solution. We call this facility $\hat{F}$. We can then show that

$$\begin{aligned}
\mathbb{E}[\mathrm{OPT}] &\leq \mathbb{E}[P(\hat{A}) + R(\hat{R})] \\
&\leq f + \frac{D}{m-1}\mathbb{E}[|V^{\hat{A}}|] + R(\hat{R}) \\
&\leq f + \frac{D}{m-1}\mathbb{E}[|V^{\hat{A}}|] + \frac{D}{\sqrt{m}}\mathbb{E}[|V^{\hat{R}}|] \\
&= f + \frac{D}{m-1}\frac{h\sqrt{m}-h}{\sqrt{m}} + \frac{D}{\sqrt{m}}\frac{h}{\sqrt{m}} \\
&\leq f + 2\frac{hD}{m-1}
\end{aligned}$$

The first inequality follows from optimality. From Fact 2, it follows that for each location $v_i \in V^{\hat{A}}$, the service cost for all the customers at v_i to $\hat{F}$ is $m^{i-1}\frac{D}{m^{i-1}(m-1)} = \frac{D}{m-1}$, which implies the second inequality. For each location $v_i \in V^{\hat{R}}$, the rejection cost for all the customers at v_i will be $m^{i-1}\frac{D}{m^{i-1}\sqrt{m}} = \frac{D}{\sqrt{m}}$, which implies the third inequality. The fourth line follows from Step 3 of the customer sequence construction and the fact that there are h levels in total(probability of $v_i \in V^{\hat{A}}$ is $1 - 1/\sqrt{m}$ for all i). The last inequality follows from simple algebra.

We now focus on the cost of an arbitrary deterministic algorithm, ALG, on this input. Let us again refer to $\mathcal{A}$ and $\mathcal{R}$ as the accepted and rejected customers by ALG. Let $V^{\mathcal{A}} \subseteq \{v_1, \ldots, v_h\}$ be the locations where at least half of the customers are in $\mathcal{A}$

and let $V^{\mathcal{R}}$ be the remaining locations. The expected cost of ALG is then

$$\begin{aligned}
\mathbb{E}[\text{ALG}] &= \mathbb{E}[P(\mathcal{A}) + R(\mathcal{R})] \\
&\geq \mathbb{E}[P(\mathcal{A} \cap \hat{\mathcal{R}})] + \mathbb{E}[R(\mathcal{R} \cap \hat{\mathcal{A}})] \\
&\geq \min\{f, D/2\}\mathbb{E}[|V^{\mathcal{A}} \cap V^{\hat{\mathcal{R}}}|] + \mathbb{E}[R(\mathcal{R} \cap \hat{\mathcal{A}})] \\
&\geq \min\{f, D/2\}\mathbb{E}[|V^{\mathcal{A}} \cap V^{\hat{\mathcal{R}}}|] + \frac{D}{2\sqrt{m}}\mathbb{E}[|V^{\mathcal{R}} \cap V^{\hat{\mathcal{A}}}|] \\
&\geq \min\{f, D/2\}\frac{\mathbb{E}[|V^{\mathcal{A}}|]}{\sqrt{m}} + \frac{D}{2\sqrt{m}}\frac{\sqrt{m}E[|V^{\mathcal{R}}|] - E[|V^{\mathcal{R}}|]}{\sqrt{m}} \\
&\geq \min\{f, D/4\}\frac{\mathbb{E}[|V^{\mathcal{A}}|]}{\sqrt{m}} + \frac{D}{4\sqrt{m}}E[|V^{\mathcal{R}}|]
\end{aligned}$$

The first inequality follows from monotonicity of $P(\cdot)$ and $R(\cdot)$. Note that to serve the customers in $\mathcal{A}$, we need to serve at least half of the customers at each location $v_i \in V^{\mathcal{A}} \cap V^{\hat{\mathcal{R}}}$. By construction of $V^{\hat{R}}$, none of the subtrees of the nodes in $V^A \cap V^{\hat{R}}$ intersect. Thus, for each v_i we must either construct a facility in T_{v_i} or serve the customers from another facility which, by Fact 1, will cost $\frac{m^{i-1}}{2}\frac{D}{m^{i-1}} = \frac{D}{2}$. Thus, the second inequality follows from the lower bound just derived of serving the accepted customers at $V^A \cap V^{\hat{R}}$. The third inequality follows from the fact that we need to reject at least half of the customers located at $V^R \cap V^{\hat{A}}$. For a given location, this will cost $\frac{m^{i-1}}{2}\frac{D}{m^{i-1}\sqrt{m}} = \frac{D}{2\sqrt{m}}$. The fourth inequality follows from the fact that the probability that any node v_i is in $V^{\hat{A}}$ is $1 - 1/\sqrt{m}$ by Step 3 of the construction, regardless of the fact that $v_i \in V^R$ as well. The last inequality follows when $m \geq 4$.

Now we will carefully set $h := m$ and $D := 4f$. Then the competitive ratio of ALG on this instance will be

$$\begin{aligned}
\frac{\mathbb{E}[\text{ALG}]}{\mathbb{E}[\text{OPT}]} &\geq \frac{\min\{f, D/4\}\frac{\mathbb{E}[|V^A|]}{\sqrt{m}} + \frac{D}{4\sqrt{m}}E[|V^R|]}{f + 2\frac{hD}{m-1}} \\
&= \frac{\frac{\mathbb{E}[|V^A|]}{\sqrt{m}} + \frac{1}{\sqrt{m}}E[|V^R|]}{1 + 8\frac{m}{m-1}} \\
&= \frac{\sqrt{m}}{1 + 8\frac{m}{m-1}} \\
&= \Omega(\sqrt{m})
\end{aligned}$$

The first inequality follows directly from the two bounds we derived above. The first equality follows from canceling out f terms. The second equality follows from the fact that the expected total number of locations is m by construction.

Finally, we need to determine m. Since there are $1 + m + \ldots + m^{m-1} \approx m^m$ total units of demand, then the most m can be is $\Theta(\frac{\log N}{\log \log N})$. Thus, the competitive ratio for any algorithm is at least $\Omega\left(\sqrt{\frac{\log N}{\log \log N}}\right)$. □

2.10 Conclusion

In this chapter, we introduced a general framework for online customer selection problems. These problems are important in industries where customers must be either explicitly or implicitly rejected. We showed that a relatively simple approach, named the Copycat Algorithm, which mimics the optimal offline solution in each stage resulted in strong worst case guarantees for several supply chain models. Then we proposed the more sophisticated StablePair Algorithm, which could be implemented in polynomial time and resulted in stronger guarantees. We provided a third algorithm called FairShare for this class of problems that is based on cost sharing mechanisms developed for cooperative games. These cost sharing mechanisms aim to naturally distribute the cost of production to the "players" in the cooperative game in such a way that the allocation is fair and recovers most of the cost. FairShare provides poly-logarithmic competitive ratio guarantees for a variety of problems with online customer selection with no restriction on the rejection cost structure. We also showed that these guarantees are close to best possible by constructing customer sequences that result in poly-logarithmic competitive ratios for all algorithms. We believe that the problems and algorithms generated in this paper might have applications in other areas.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Retailing with Opaque Products

3.1 Introduction

In this chapter, we study the potential impact of offering *opaque products* in online retail environments. An opaque product is defined as a product where the consumer only learns of all the product's attributes after the purchase has been made. In the context we consider, when a customer purchases an opaque product he or she agrees to accept any product from a pre-specified list of products. Once the purchase of an opaque product is complete, the seller decides which one of the products from the pre-specified list to give to the customer. Opaque products have been almost exclusively used and studied in the travel industry, where consumers can buy airline tickets, hotel rooms, and car rentals without knowing exactly which company they will receive, as well as possibly the type of flight, room, or car. Priceline.com and Hotwire.com are two of the major third-party channels for selling opaque products. Opaque selling allows airlines to offer lower priced airfare to customers via third party channels, without cannibalizing the original posted prices. Thus, opaque products allow airlines to take advantage of price differentiation and increase overall revenues, with a percentage of the opaque sales given to the third-party channel.

The goal of our work is to show that selling opaque products in the retail industry could be beneficial from both a revenue *and* cost perspective. For example, an opaque product may be a household item with two different color possibilities, and the consumer only learns the color they will receive after the purchase is complete (upon delivery). Based on our findings in this work, we believe that online retailers can and should implement the sales of opaque products, a practice that currently exists in very limited applications to the best of our knowledge. One major difference is that we propose that online retailers sell opaque products directly *alongside* transparent (original) products. Specifically, we focus on the scenario where the transparent products are highly substitutable and have the same basic functionality, but may vary in a stylistic feature such as shape, color, or design. For an appropriate incentive, we believe that there are customers that would be willing to buy an opaque product if he is practically indifferent between the transparent options. Although this incentive could take on many forms, we will specifically focus on a price discount incentive for purchasing an opaque product.

In our work we show that selling opaque products can offer significant improvements for the retailer's supply chain costs. When a customer purchases an opaque product, the retailer can provide the product that puts him in the best inventory position afterwards. We show that this can result in significant savings on holding and ordering costs. By having customers who purchase opaque products, the retailer is essentially reducing the volatility of demand, and thus driving down costs. Specifically, we can leverage the use of opaque products to delay necessary replenishments and reduce the average inventory on hand by a significant amount. Moreover, we only require a small percentage of customers to purchase the opaque product to achieve major cost savings advantages. Our analysis relies on studying a key metric: the expected number of customers between replenishments. Based on this metric, we derive closed-form expressions for the holding and ordering cost rates as a function of the fraction of customers that purchase opaque products. Furthermore, when there are only two transparent products we provide theoretically results on these rates that are asymptotically tight.

We also address the issue of estimating the demand under an opaque selling strategy. Given a demand model for the products under a traditional selling strategy (no opaque products), and assuming customers are risk-averse, we can exactly characterize the demand under the opaque selling strategy with closed-form expressions. This characterization allows us to accurately compute the demand rates for transparent and opaque products for a given price structure, and therefore study the overall impact of an opaque selling strategy. As we will show, the optimal opaque strategy may sometimes result in less revenue but a better overall profit due to the cost savings (contrary to the travel industry). We do however show that under minor conditions, it is possible to improve both revenue and profit, which may be a desirable situation for a retailer.

We also compare our opaque selling strategy to a certain class of dynamic pricing policies for a traditional selling model. Specifically, this class of dynamic pricing policies must always sell one product at a given full price, and the other at some discount. Under some conditions, we are able to show that our opaque selling strategy can outperform this class of dynamic pricing policies. In fact, we show that the costs are identical, and the revenue of the opaque strategy is superior. We believe that this result suggests that opaque selling strategies can be used as an alternative to dynamic pricing, or at least should be considered as an alternate mechanism for managing inventory.

Finally, we also consider cases where there are more than two transparent products, in which case the cost savings from opaque selling increases even further. Moreover, we introduce a generalization of opaque selling called k-opaque products. In this setting, the customer first chooses k options, and then the retailer will choose one out of the k options. For example, if the retailer had 10 transparent options and sold 2-opaque goods, then customers choose their favorite two and then the retailer chooses one of those two. Surprisingly, our computational results suggest that 2-opaque strategies are nearly as effective as selling fully opaque products. To the best of our knowledge, we are not aware of any current practice of selling k-opaque products.

3.1.1 Literature Review

Opaque products have been primarily studied in industries with perishable inventory such as airline tickets, hotel rooms, and cruise ships ([49]). [29] described the notion of a probabilistic good, which is similar to an opaque product except that the retailer specifies the probability of receiving each traditional product. [50] analyze opaque selling in a two-stage model with competition, and compare this to last-minute selling strategies. [35] looks at revenue management of flexible products, which is when the details of the product are revealed to the consumer close to the time of consumption (i.e. the day before a reservation).

Our work is fundamentally different than previous research as we focus on nonperishable retail goods and the impact that selling opaque products has on supply chain costs, in addition to revenue. Although previous research focused on pricing issues, here opaque sales may be achieved by other means such as rewards points, preferred shipping methods, or even no incentives for consumers who dislike choices and enjoy surprises. However, we also contribute a new way to estimate opaque demand levels when customers respond to price incentives, which may also be useful for traditional applications of opaque products.

Our work is also related with multi-item inventory-pricing problems with joint replenishment costs and substitutable demand. Due to the so-called curse of dimensionality, it has been well documented that even for simple multi-item inventory problems without pricing and substitutable demand, optimal policies are prohibitively difficult to find and may not yield simple solution form (see [3] and [45]). Therefore, in our work, we fix our inventory policy and propose a simple model that captures the cost of inventory imbalance caused by the stochastic nature of the demand, and use the propose model to analyze the benefit of opaque selling. For computing optimal inventory-pricing policies under production or inventory ordering systems, we refer the readers to [81], [96], and [18].

This chapter proceeds as follows. In Section 3.2, we present the opaque selling model and a preliminary analysis for the case of two transparent products with symmetric demand. In Section 3.3, we analyze in depth the supply chain cost savings from a computational and theoretical perspective. In Section 3.4, we derive a method to estimate demand and apply the techniques to a linear demand model for substitutable goods. In Section 3.5, we compare our opaque selling strategy to a class of dynamic pricing policies. Finally, in Section 3.6 we discuss extensions to our model that include asymmetric demand and k-opaque products.

3.2 The Opaque Product Model

Model Dynamics. In our model, we consider an online retailer selling non- perishable products over an infinite time horizon. The retailer offers two types of substitutable products, product 1 and product 2, at prices p_1 and p_2 respectively. We assume that these products come from the same supplier and vary in a stylistic feature such as color or shape. Unless specified otherwise (e.g., Section 3.6.1), we will assume that the demand and prices for products 1 and 2 are identical, and thus we will refer to their price as $p = p_1 = p_2$. In addition to offering products 1 and 2 for sale, the retailer offers an *opaque* option, which means a consumer will receive either product 1 or product 2, at the retailer's discretion. The price of the opaque product is $p - \delta$, where the discount δ rewards the customer for sacrificing his choice. For clarity purposes, we will refer to products 1 and 2 as *transparent* products, and the opaque option as an opaque product. Furthermore, we refer to this model as an *opaque selling* strategy. If the retailer sells only transparent products, we will refer to it as a *traditional selling* strategy.

For a given p and δ, we model the demand by a stochastic arrival process with independent and identically distributed inter-arrival times with mean $1/\lambda$. Alternative, λ can be thought of as the demand per time unit. When a customer arrives, he purchases a transparent product with probability $1 - q$, and an opaque product with probability q. Since the demand and prices for products 1 and 2 are identical, when a customer purchases a transparent product he chooses from products 1 and

2 uniformly at random, each with probability 1/2. When a customer chooses the opaque product, the retailer can fulfill the purchase with either product 1 or product 2. Note that both the demand rate λ and the opaque selection probability q are dependent on the price p of the transparent products and opaque product discount δ. When necessary, we will use $\lambda(p, \delta)$ and $q(p, \delta)$ to denote these dependencies. When the demand rate, opaque selection probability, and prices are known, the revenue per time unit can be expressed as

$$\lambda((1-q)p + q(p-\delta)).$$

To satisfy demand, the retailer continuously orders products 1 and 2 from its supplier and satisfies the demand from its on-hand inventory. For each order, there is a joint ordering cost K for ordering (replenishing) inventory, which is motivated by the fact that the products are from the same supplier. Moreover, there is a holding cost H for holding one item for one time unit. For simplicity, we assume that there is no order lead time, backlogging, or lost sales. This assumption implies that under the optimal inventory policy, an order is triggered immediately when the inventory level for any product drops to zero. Every time a replenishment order occurs, the inventory of each product is replenished up to a reorder level c. One can think of c as either the retailer's maximum capacity for each product, or the reorder level chosen by the retailer. (Note that here we assume the inventory policy will always remain fixed, although in practice one may gain further savings by reoptimizing the inventory policy under an opaque selling strategy.)

Fulfillment of the Opaque Product. Under the opaque product model we have described, we can characterize retailer's optimal fulfillment strategy when a customer chooses the opaque product. Due to the demand symmetry between products 1 and 2, intuitively the retailer should always attempt to have a balanced inventory position to prepare himself for future demand. This is exactly what the fulfillment strategy in Theorem 1 (proof in Appendix C.1.1) is trying to accomplish when a customer buys an opaque product.

Proposition 1. *Under symmetric demand for transparent products, it is always optimal for the retailer to fulfill an opaque product request with the transparent product that has the highest on-hand inventory at the time.*

In the rest of the chapter, we will assume this optimal fulfillment strategy is always applied when an opaque product is sold.

3.2.1 Computing the Inventory Cost

Next, we analyze the retailer's average cost rate under our inventory model and demonstrate that the cost can be expressed as a function of the expected number of customers served between consecutive replenishments. We use the random variable $R(q, c)$ to denote the number of customers served between consecutive replenishments when the fraction of opaque customers is q and the reorder level is c. For example, $R(1, c)$ represents the fully opaque setting where all customers purchase the opaque product, while $R(0, c)$ represents the traditional setting where no customers purchases the opaque product. Note that $R(1, c) = 2c - 1$ with probability 1, since the retailer will alternate between products 1 and 2 until the inventory of one product reaches zero, at which there must be 1 unit of inventory left for the other product and a replenishment order is triggered.

Our analysis of the inventory costs will heavily rely on the expected value of the number of customers served between consecutive replenishments, i.e., $\mathbb{E}[R(q, c)]$. Theorem 2 (proof in Appendix C.1.2) characterizes the ordering costs and holding costs in terms of this quantity. The proposition also shows how much savings we get compared to the traditional strategy where there is no opaque selling, i.e. $q = 0$. In addition, we also describe the relative savings achieved when comparing to a fully opaque setting ($q = 1$), which is a "best case" scenario where we can save the most in supply chain costs.

Proposition 2. *For any $0 \leq q \leq 1$ and integer $c \geq 1$,*

(a) the ordering cost rate is $\frac{\lambda K}{\mathbb{E}[R(q,c)]}$.

(b) *the ordering cost savings compared to the traditional setting (no opaque) is* $1 - \frac{\mathbb{E}[R(0,c)]}{\mathbb{E}[R(q,c)]}$.

(c) *the relative ordering cost savings compared to all opaque selling is* $\frac{\mathbb{E}[R(1,c)](\mathbb{E}[R(q,c)-R(0,c)])}{\mathbb{E}[R(q,c)](\mathbb{E}[R(1,c)-R(0,c)])}$.

(d) *the holding cost rate is* $\frac{(4c-\mathbb{E}[R(q,c)]+1)H}{2}$.

(e) *the holding cost savings compared to the traditional setting (no opaque) is* $1 - \frac{4c-\mathbb{E}[R(q,c)]+1}{4c-\mathbb{E}[R(0,c)]+1}$.

(f) *the relative holding cost savings compared to all opaque selling is* $\frac{\mathbb{E}[R(q,c)-R(0,c)]}{\mathbb{E}[R(1,c)-R(0,c)]}$.

Theorem 2 implies that once the value of $\mathbb{E}[R(q,c)]$ is known, it is easy to compute retailer's costs in both absolute and relative terms.Interestingly, while our model is continuous time, $R(q,c)$ can be computed by analyzing a simple discrete-time 2-dimensional Markov chain with $O(c^2)$ states. Each state corresponds to the inventory remaining of the two products. If $i \geq j \geq 2$, the probability of moving from state (i,j) to $(i,j-1)$ is $(1-q)/2$ and to $(i-1,j)$ is $(1-q)/2+q$. Any state with $i=0$ or $j=0$ is an absorbing state, and $\mathbb{E}[R(q,c)]$ is equivalent to the expected absorption time if we start at state (c,c). The value of $\mathbb{E}[R(q,c)]$ can also be computed fairly easily via simulation, which may be preferable under more general models.

Finally, we note that parts (b),(c),(e), and (f) of Theorem 2 not only allow us to compute the savings of different opaque levels, but also demonstrate that these relative savings are independent of the cost parameters and the demand rate. In the subsequent sections, this property would allow us to analyze the potential cost savings and obtain generally applicable insights without the exact values of the cost parameters.

3.2.2 Profitability of Opaque Selling

In this subsection, we analyze the benefit of moving from the traditional selling model to the opaque selling model. In particular, we will show that moving from a traditional selling strategy to an opaque selling strategy is always beneficial, for the appropriate choice of δ. We will consider a retailer under the traditional selling model with prices for both transparent products equal to p, and analyzes the retailer's change in profit when it introduces an opaque product with discount $\delta \geq 0$. This is motivated by a retailer's potential unwillingness or inability to change p, but would consider selling an opaque product. We will prove that even if the retailer does not change the price of the transparent products, there always exists a discount $\delta \geq 0$ such that the opaque selling enables the retailer to strictly improve its profit under very general conditions. We will also show that under some conditions, the retailer may not even need to sacrifice revenue in order to achieve this improvement.

Without loss generality, we assume in this subsection that $p = 1$. For notational simplicity, we use $q(\delta)$ to replace $q(1, \delta)$ and $\lambda(\delta)$ to replace $\lambda(0, \delta)$. Note that if the customers behave rationally, then the customer arrival rate should not decrease for any positive discount offered at the opaque product. Therefore, throughout the section, we assume $\lambda(\delta) \geq \lambda(0)$, $\forall \delta \geq 0$. Moreover, to avoid unnecessary derivations under unrealistic settings, we assume $\lambda(0)$ is equal to the customer arrival rate under the traditional selling model, $\lambda(0) > 0$, and $q(0) = 0$. (Note that if $q(0) > 0$, then our profit improves for free.)

The profit rate for an opaque selling strategy with discount δ will be denoted by $\Pi(\delta)$, which can be expressed as:

$$\Pi(\delta) := \lambda(\delta)(1 - q(\delta)\delta) - \frac{\lambda(\delta)K}{\mathbb{E}[R(q(\delta), c)]} - \frac{(4c - \mathbb{E}[R(q(\delta), c)] + 1)H}{2}, \tag{3.1}$$

where the first part of the equation represents the revenue rate, while the second and third part of the equation represent the ordering and holding cost rates.

Next, we demonstrate that under some mild conditions, a retailer changing from the traditional selling model to the opaque selling model will always receive a strict increase in profit if the opaque product discount is chosen correctly. We also study the impact of the opaque product to the retailer's revenue. Note that while the opaque product may increase the overall demand rate, it may cause the revenue to decrease if too many customers are choose the opaque product over the transparent products.

Proposition 3. *Suppose the right derivative of $q(\delta)$ at 0 exists and is strictly positive, the following results hold:*

(a) There always exists some $\delta_0 > 0$ such that when the opaque product is introduced at discount δ_0, the increase in retailer's profit is strictly greater than 0.

(b) Suppose in addition, the right derivative of $\lambda(\delta)$ at 0 exists and is strictly positive. Then there exists some $\delta_0 > 0$ such that when the opaque product is introduced at discount δ_0, the increase in both retailer's profit and revenue are strictly greater than 0.

The proof of Theorem 3 can be found in Appendix C.1.3. We note that the conditions where the right derivatives of $q(\delta)$ and $\lambda(\delta)$ is strictly positive essentially imply that there is a positive rate of increase in the number of opaque purchases as well as the demand rate when a small discount is offered. In the next section, we will provide an in-depth analysis on the cost, and potential savings, of an opaque selling strategy. We will show that even a small fraction of opaque sales can provide a lot of cost savings, which refines our intuition for Theorem 3. Then in Section 3.4, we will show how to estimate $\lambda(p, \delta)$ and $q(p, \delta)$ and we will study in-depth the profit of an opaque selling strategy for a particular demand model.

3.3 Inventory Cost Savings with Opaque Selling

In this section, we focus on the costs of an opaque selling strategy. For this purpose, we will assume that λ remains unchanged for different values of q, the opaque selection probability. These results will allow us to describe the value of converting q fraction of the demand from a traditional model into opaque customer. We will study the cost savings for different values of q and c. Note that by Theorem 2, the relative ordering and holding cost savings are independent of λ, K, and H.

The analysis of the cost savings in the section is divided into two subsections. In Section 3.3.1, we provide some numerical examples to demonstrate the cost advantages of the opaque product and develop some intuitions. In Section 3.3.2, we develop several theoretical properties on the cost savings of the opaque product, which allows us to refine the intuition developed in Section 3.3.1 and provide important insights.

3.3.1 Numerical Results

We now provide numerical evidence that describes the cost advantages of opaque selling. In the next two figures, we compute the ordering cost savings and holding cost savings for the model described in Section 3.2, for various opaque levels q and reorder levels c. The savings refer to the amount of money saved in comparison to a traditional setting where there are no opaque sales ($q = 0$) and λ is the same. These computations can be computed via simulation or more direct methods as described in the previous section.

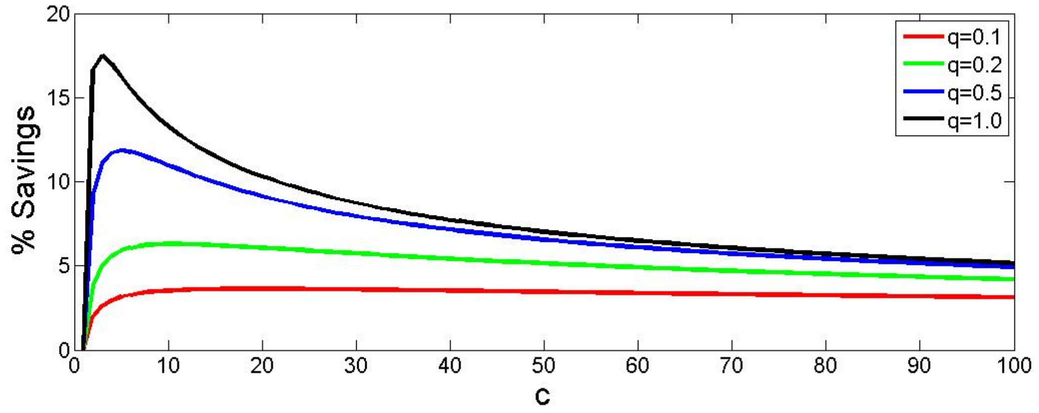

Figure 3-1: We compute the ordering cost savings achieved for different values of q and c, assuming that λ remains fixed.

From Figure 3-1, we observe several interesting facts. First, in an ideal setting when all customers purchase opaque products, i.e. $q = 1$, the retailer can save at most 18% in ordering costs, and up to 5% savings for large values of the reorder level c. However, when only 10% of customers purchase opaque products, the savings is up to 3%, even for large values of c. This implies that there is a nonlinear relationship between the cost savings and the opaque level q. This nonlinearity is advantageous since it implies that only a small fraction of customers need to purchase the opaque

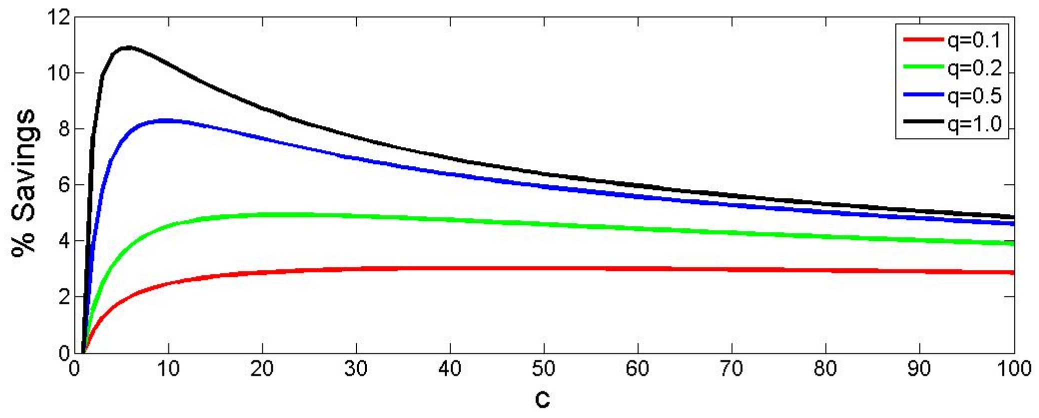

Figure 3-2: We compute the holding cost savings achieved for different values of q and c, assuming that λ remains fixed.

product in order to achieve significant cost savings. In Theorem 5, we analytically explain this behavior. From Figure 3-1, we can also observe that the savings all approach 0 as c goes to ∞, which we can also prove in the next section. Note however that the convergence is very slow, and for very reasonable and practical levels of c there are significant cost savings. A similar analysis also holds for the holding cost savings depicted in Figure 3-2. Remember that the holding costs are reduced because opaque products sales result in less on-hand inventory when orders occur, and therefore less average on-hand inventory.

3.3.2 Theoretical Analysis

To enhance the observations made in Section 3.3.1, we now analytically compare the traditional selling (no opaque) strategy versus the opaque selling strategy for different values of q. For our analysis, we will make use of the following definitions. Let $n_t^i(q)$ be the number of product i purchased given t purchasing customers have arrived since the last replenishment. Let $M_t(q)$ be the absolute value of the difference between the amount of product 1 and the amount of product 2 consumed after t customers have arrived, i.e, $M_t(q) := |n_t^1(q) - n_t^2(q)|$. Note that $n_t^i(q)$ and $M_t(q)$ are discrete time stochastic processes. For the sake of succinctness, when q and c are fixed without ambiguity, we will sometimes use R, n_t^i and M_t in place of $R(q, c)$, $n_t^i(q)$, and $M_t(q)$.

First, we show that $\mathbb{E}[R(1,c) - R(0,c)]$ is on the order of $\sqrt{c}$, implying that we serve on the order of $\sqrt{c}$ more customers between replenishments when all customers purchase the opaque product.

Proposition 4. *As the reorder level c goes to infinity,* $\frac{\mathbb{E}[R(1,c)-R(0,c)]}{\sqrt{2c}}$ *approaches* $\sqrt{\frac{2}{\pi}}$.

In the proof of Theorem 4, we first show that $\mathbb{E}[R(1,c) - R(0,c)]$ can be rewritten as the expected absolute value of a sum of $2c$ IID random variables. Then we scale $\mathbb{E}[R(1,c) - R(0,c)]$ by $\sqrt{2c}$ and apply the Central Limit Theorem together to obtain the limit as c approaches infinity. The formal proof is provided in Appendix C.1.4.

Theorem 4 provides us with a formula to accurately estimate $\mathbb{E}[R(1,c) - R(0,c)]$ when the re-order level c is large. Moreover, with Theorem 4, we have

$$1 - \frac{\mathbb{E}[R(0,c)]}{\mathbb{E}[R(q,c)]} \leq \frac{\mathbb{E}[R(1,c) - R(0,c)]}{\mathbb{E}[R(1,c)]} \to 0, \tag{3.2}$$

$$\text{and } 1 - \frac{4c - \mathbb{E}[R(q,c)] + 1}{4c - \mathbb{E}[R(0,c)] + 1} \leq \frac{\mathbb{E}[R(1,c)] - R(0,c)]}{4c - \mathbb{E}[R(0,c)] + 1} \to 0, \tag{3.3}$$

as $c \to \infty$. Recall that from Theorem 2, $1 - \frac{\mathbb{E}[R(0,c)]}{\mathbb{E}[R(q,c)]}$ and $1 - \frac{4c - \mathbb{E}[R(q,c)]+1}{4c - \mathbb{E}[R(0,c)]+1}$ represent the ordering and holding cost savings relative to the costs under the traditional setting. Therefore, Equations 3.2 and 3.3 implies that for any $0 < q \leq 1$, the value of opaque selling converges to 0 as c gets large. However, since the convergence is on the order of $\frac{1}{\sqrt{c}}$, the value of opaque selling goes to 0 at an extremely slow rate. Indeed, in Figures 3-1 and 3-2, one can see that the ordering and holding cost savings is significantly above 0 for large values of c.

Next, we would like to understand how much of the relative savings are captured with q fraction of our customers being opaque compared to the ideal setting when all customers are opaque (see parts (c) and (f) of Theorem 2). From the computational results, one can see that there is a nonlinear relationship between q and the relative savings captured, and we aim to describe that behavior in this section.

To analyze the relative savings, we first state a lemma which establishes the relationship between $R(q,c)$ and $M_t(q)$.

Lemma 22. *Fix q and c. For any positive integer $c \leq x \leq 2c$, $\mathbb{P}[R(q,c) \leq x] = \mathbb{P}[M_x(q) \geq 2c - x]$. Equivalently, we have that for any positive integer $1 \leq d \leq c$, $\mathbb{P}[2c - R(q,c) \geq d] = \mathbb{P}[M_{2c-d}(q) \geq d]$.*

In Theorem 5 below, we prove that for any $q > 0$, the difference between the expected replenishment time of fully opaque demand and the expected replenishment time of fractional opaque demand can be uniformly bounded for any re-order level c. More specifically, our depends solely on the value of q. This will help us provide a simple lower bound for $\mathbb{E}[R(q,c)]$, as well as to estimate the relative savings of opaque selling compared to an ideal situation where all customers purchase opaque products.

Proposition 5. *For any $0 < q \leq 1$, $\mathbb{E}[R(1,c) - R(q,c)] \leq \frac{1-q}{2q}$. Equivalently, $\mathbb{E}[R(q,c)] \geq 2c - 1 - \frac{1-q}{2q}$.*

Proof. Throughout the proof, we will fix some arbitrary $q > 0$, and use M_x to denote $M_x(q)$. Recall that $R(1,c) = 2c - 1$ with probability 1. By Lemma 22, and the fact that $0 \leq R(1,c) - R(q,c) \leq c - 1$,

$$\begin{aligned}
\mathbb{E}[R(1,c) - R(q,c)] &= -1 + \mathbb{E}[2c - R(q,c)] \\
&= -1 + \sum_{d=1}^{c} \mathbb{P}[2c - R(q,c) \geq d] \\
&= -1 + \sum_{d=1}^{c} \mathbb{P}[M_{2c-d} \geq d]. \qquad (3.4)
\end{aligned}$$

Now observe that $\mathbb{P}[M_{i+1} = 1 | M_i = 0] = 1$, $\mathbb{P}[M_{i+1} = M_i - 1 | M_i > 0] = \frac{1}{2}(1+q)$, and $\mathbb{P}[M_{i+1} = M_i + 1 | M_i > 0] = \frac{1}{2}(1-q)$. Since M_i is periodic, we will decompose it into two aperiodic Markov chains N and N'. We define the stochastic processes $\{N_i := M_{2i} : i = 0, 1, ...\}$ and $\{N'_i := M_{2i-1} : i = 1, 2, ...\}$. In particular, $\{N_i : i = 0, 1, ...\}$ is the Markov chain such that

$$\begin{aligned}
\mathbb{P}[N_{i+1} = 2 | N_i = 0] &= \frac{1}{2}(1-q), \\
\mathbb{P}[N_{i+1} = 0 | N_i = 0] &= \frac{1}{2}(1+q) \\
\mathbb{P}[N_{i+1} = N_i - 2 | N_i > 0] &= \frac{(1+q)^2}{4}, \\
\mathbb{P}[N_{i+1} = N_i + 2 | N_i > 0] &= \frac{(1-q)^2}{4} \\
\mathbb{P}[N_{i+1} = N_i | N_i > 0] &= 1 - \frac{(1+q)^2}{4} - \frac{(1-q)^2}{4} = \frac{1-q^2}{2}.
\end{aligned}$$

We note that $\{N_i : i = 0, 1, ...\}$ is aperiodic, and has a steady state probability vector $\boldsymbol{\pi}$ computed in Appendix C.2. We will use N_∞ to denote the steady state distribution of $\{N_i; i = 0, 1, 2, ...\}$.

Moreover, the transition probabilities of $\{N_i : i = 0, 1, ...\}$ preserves stochastic dominance (a rigorous proof on this result can be found in Appendix C.1.6). More specifically, given two random variables D and D' with non-negative integer supports such that $\mathbb{P}[D \geq k] \geq \mathbb{P}[D' \geq k]$ for all integers $k \geq 0$, and random variables $X := (N_1 | N_0 \overset{d}{=} D)$ and $Y := (N_1 | N_0 \overset{d}{=} D')$, we must have that $\mathbb{P}[X \geq k] \geq \mathbb{P}[Y \geq k]$.

Due to the preservation of stochastic dominance, then we have that $N_\infty = (N_i | N_0 \overset{d}{=} N_\infty)$ stochastically dominates $(N_i | N_0 = 0)$. Therefore,

$$\begin{aligned}
\sum_{k=1}^{c/2} \mathbb{P}[M_{2c-2k} \geq 2k] &= \sum_{k=1}^{c/2} \mathbb{P}[N_{c-k} \geq 2k | N_0 = 0] \\
&\leq \sum_{k=1}^{\infty} \mathbb{P}[N_{c-k} \geq 2k | N_0 = 0] \\
&\leq \sum_{k=1}^{\infty} \mathbb{P}[N_\infty \geq 2k] \\
\text{(since } \mathbb{P}[N_\infty \geq 2k] = \mathbb{P}[N_\infty \geq 2k-1]\text{)} \qquad &= \frac{1}{2} \sum_{k=1}^{\infty} \mathbb{P}[N_\infty \geq k] \\
&= \mathbb{E}[N_\infty]/2 \\
&= \frac{(1-q)(1+q)}{4q}
\end{aligned}$$

where the last equality is derived by computing the expected state of the Markov chain N_i (shown in Appendix C.2).

Now consider the stochastic process $\{N'_i := M_{2i-1} : i = 1, 2, ...\}$. Like $\{N_i : i = 0, 1, ...\}$, $\{N'_i := M_{2i-1} : i = 1, 2, ...\}$ is aperiodic, positive recurrent and preserves stochastic dominance. Let $\mathbb{E}[N'_\infty]$ be the steady state distribution of (N'_i), and we have

$$\begin{aligned}
\sum_{k=1}^{c/2} \mathbb{P}[M_{2c-2k+1} \geq 2k-1] &= \sum_{k=1}^{c/2} \mathbb{P}[N'_{c-k+1} \geq 2k-1 | N'_1 = 1] \\
&\leq \sum_{k=1}^{\infty} \mathbb{P}[N'_{c-k+1} \geq 2k-1 | N_0 = N_\infty] \\
&= \sum_{k=1}^{\infty} \mathbb{P}[N'_\infty \geq 2k-1] \\
\text{(since } \mathbb{P}[N'_\infty \geq 2k+1] = \mathbb{P}[N'_\infty \geq 2k]) \qquad &= \frac{1}{2}\sum_{k=2}^{\infty} \mathbb{P}[N'_\infty \geq k] + \mathbb{P}[N'_\infty \geq 1] \\
&= \frac{\mathbb{E}[N'_\infty]}{2} + \frac{\mathbb{P}[N'_\infty \geq 1]}{2} \\
&= \frac{1+q^2+2q}{4q},
\end{aligned}$$

Combining the inequalities on $\sum_{k=1}^{c/2} \mathbb{P}[M_{2c-2k} \geq 2k]$ and $\sum_{k=1}^{c/2} \mathbb{P}[M_{2c-2k+1} \geq 2k-1]$, we have that $\sum_{k=1}^{c} \mathbb{P}[M_{2c-k} \geq k] \leq \frac{(1-q)(1+q)}{4q} + \frac{1+q^2+2q}{4q} = \frac{1+q}{2q}$. And substituting this inequality into Equation 3.4, we get $\mathbb{E}[R(1,c) - R(q,c)] = \sum_{k=1}^{c} \mathbb{P}[M_{2c-k} \geq k] - 1 \leq \frac{1-q}{2q}$. □

Combining Theorems 4 and 5, we obtain

$$\begin{aligned}
\frac{\mathbb{E}[R(q,c) - R(0,c)]}{\mathbb{E}[R(1,c) - R(0,c)]} &\geq \frac{\mathbb{E}[R(1,c) - R(0,c)] - (1-q)/2q}{\mathbb{E}[R(1,c) - R(0,c)]} \\
&= 1 - \frac{1-q}{2q\mathbb{E}[R(1,c) - R(0,c)]} \\
&\to 1 - \frac{\sqrt{\pi}(1-q)}{4q\sqrt{c}}, \text{ as } c \to \infty.
\end{aligned}$$

Therefore, by Theorem 2, if q fraction of customers purchases the opaque product, then the relative savings on both the ordering and holding cost compared to all customers buying opaque product is at least $1 - \frac{\sqrt{\pi}(1-q)}{4q\sqrt{c}}$ as c is large. While the expression is an asymptotic result, $\frac{\mathbb{E}[R(q,c)-R(0,c)]}{\mathbb{E}[R(1,c)-R(0,c)]}$ in facts converges to $1 - \frac{\sqrt{\pi}(1-q)}{4q\sqrt{c}}$ very quickly (see Figure 3-3). Therefore, $1 - \frac{\sqrt{\pi}(1-q)}{4q\sqrt{c}}$ can be used as a formula to quickly estimate the value of opaque selling, for $c > 20$.

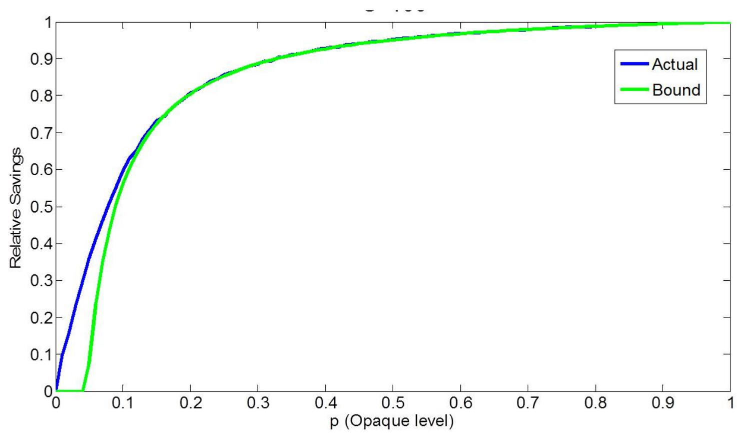

Figure 3-3: We compute the relative savings of the ordering cost when $c = 100$ and compare it to the lower bound found in Theorem 5.

The bound we derived above on the relative savings has several interesting features. First, we observe that as c gets very large, any non-negative value of opaque selling gives us the same magnitude of savings as having all of the customers buying opaque products. Moreover, the bound implies that the relative savings of opaque selling level q grows very quickly in q since the negative part goes to 0 at a rate of $\frac{1}{q}$ which is very fast. This explains why only a small fraction of opaque sales is necessary to achieve significant cost savings advantages. Therefore, these observations provide us with an important insight: when the retailer has a high reorder level, a tiny fraction of opaque customers will give him most of the possible cost savings. This implies that a large discount δ may not be necessary, or the retailer may even use other means such as reward points to incentivize some customer to purchase opaque products. In the next section, we describe in detail how to estimate the demand based on p and δ.

3.4 Estimating Demand for Opaque Selling Strategies

In the previous sections, we have focused our effort in understanding the cost savings achieved from opaque selling. However, one also needs to be concerned about ensuring that the overall opaque sales strategy is still profitable. As previous literature has shown, one can actually use opaque products as a mechanism for price differentiation and thus increase overall revenue. However, in the optimal price configuration under the opaque strategy there may be both a revenue and cost decreases, revenue and cost increases, or a revenue increase and cost decrease. Hence, it is essential to understand how to estimate opaque demand in order to evaluate the overall revenue and profit of opaque selling.

We will again focus on the scenario with two transparent products, namely product 1 and product 2, and an opaque product. If we sell products 1 and 2 at an arbitrarily fixed price p and the opaque product at price $p - \delta$, then we would like to estimate the arrival rates for each product, namely $\lambda_1(p, \delta)$, $\lambda_2(p, \delta)$, and $\lambda_o(p, \delta)$. The opaque fraction can then be computed as $q(p, \delta) = \lambda_o(p, \delta)/(\lambda_1(p, \delta) + \lambda_2(p, \delta) + \lambda_o(p, \delta)$.

3.4.1 Customer Choice Model

We will now describe a choice model for how customers make decisions. Specifically, we assume each customer has a (possibly different) valuation v_1 for product 1 and v_2 product 2, although the distribution of (v_1, v_2) is unknown. The utility for purchasing follows a traditional quasi-linear behavior. A customer gets a utility of $v_1 - p$ for buying product 1, $v_2 - p$ for buying product 2, and 0 for no purchase. When an opaque product is available, we assume that customers are *risk-averse* and will value the opaque product at $\min(v_1, v_2)$. If they purchase the opaque product, they receive a utility of $\min(v_1, v_2) - (p - \delta)$. Customers will purchase the product that provides the highest nonnegative utility, or make no purchase at all if no such product exists. Our assumption of risk-averseness may represent customer behavior well if the retailer does not give any guarantee on which product will be provided. In this setting, customers are likely to assume the worst when deciding whether or not to buy an opaque product in order to make robust decisions. Even if the retailer does guarantee an equal probability, many customers may be unwilling to trust the retailer.

In Figure 3-4, we provide a useful illustration of the domain of all possible valuations that a customer may have, although the actual probability distribution over this area is unknown. In the figure, we have named six regions of importance. For example, region I refers to all customers with valuations in $[p-\delta, p] \times [0, p-\delta]$.

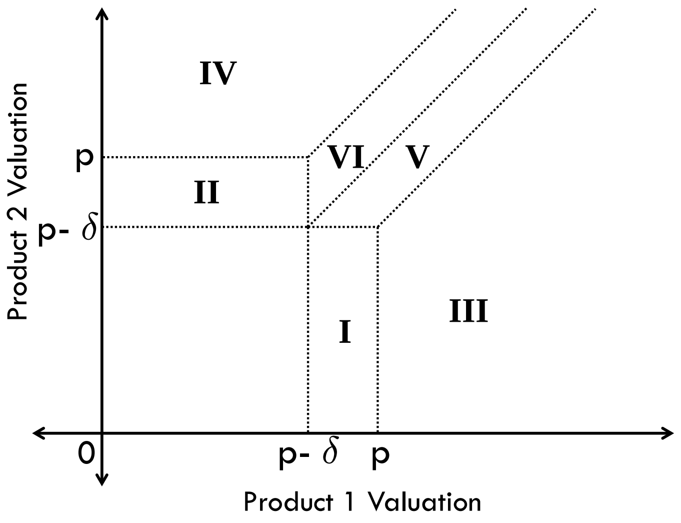

Figure 3-4: This graph represent all possible valuations for customers. We have divided up the first quadrant into different segments to be used in our analysis.

Under this choice model, we can exactly characterize the customers corresponding to $\lambda_1(p,\delta)$, $\lambda_2(p,\delta)$, and $\lambda_o(p,\delta)$ using Figure 3-4. Specifically, $\lambda_1(p,\delta)$ corresponds to the customers in region III, $\lambda_2(p,\delta)$ corresponds to region IV, and $\lambda_o(p,\delta)$ corresponds to regions V and VI combined. To see this, first note that any customer who does not value either product more than $p-\delta$ will not purchase anything. Moreover, customers in regions I and II will not purchase anything because the transparent products are too expensive, and the opaque product will result in a negative utility due to their risk averseness. However, customers who have utilities in $[p-\delta, p] \times [p-\delta, p]$ will purchase the opaque product since they are guaranteed a nonnegative utility. The boundary between regions III and V corresponds to the indifference between purchasing product

1 and the opaque product. Specifically, this indifference occurs when $v_1 > v_2$ and $v_1 - p = v_2 - (p - \delta)$. A similar analysis holds for the boundary between regions IV and VI.

Figure 3-4 can also be used to represent the demands of products 1 and 2 under a traditional selling model without opaque products, which we will use in our estimate of the demand rates under opaque selling. Namely, define $d_1(p_1, p_2)$ and $d_2(p_1, p_2)$ to be the demand rates of products 1 and 2 when sold at prices p_1 and p_2. From the figure, we can see that $d_1(p, p - \delta)$ corresponds to the customers in region III. Specifically, all customers in region III prefer to buy product 1 over product 2 at prices p and $p - \delta$, respectively, and the boundary between regions III and V is where consumers are indifferent between product 1 and 2. Customers with $v_1 \leq p$ and $v_2 \leq p - \delta$ will not purchase at all, and the remaining customers in regions II, IV, V, and VI will purchase product 1. Table 3.1 summarize other useful demand characterizations at different price points.

Prices (p_1, p_2)	$d_1(p_1, p_2)$	$d_2(p_1, p_2)$
$(p, p - \delta)$	III	II, IV, V, VI
$(p - \delta, p)$	I, III, V, VI	IV
$(p - \delta, p - \delta)$	I, III, V	II, IV, VI

Table 3.1: For each pair of prices (no opaque product), we describe the regions that correspond to demand for products 1 and 2.

From Table 3.1, we can actually exactly compute $\lambda_1(p, \delta)$, $\lambda_2(p, \delta)$ and $\lambda_o(p, \delta)$ if we know the demand arrival rates, under the traditional selling model, at price points $(p, p - \delta)$, $(p - \delta, p)$, and $(p - \delta, p - \delta)$. For example, $d_1(p, p - \delta)$ corresponds exactly to the region of customers represented by $\lambda_1(p, \delta)$. The complete characterization is stated formally in the proposition below, which is trivial to prove using Table 3.1 and our previous description of the regions that correspond to $\lambda_1(p, \delta)$, $\lambda_2(p, \delta)$ and $\lambda_o(p, \delta)$.

Proposition 6. *Assume customers are risk averse and the demand functions $d_1(\cdot,\cdot)$ and $d_2(\cdot,\cdot)$ are known at price points $(p, p-\delta)$, $(p-\delta, p)$, and $(p-\delta, p-\delta)$. Under the opaque selling model with a discount δ, the arrival rates are exactly*

- $\lambda_1(p,\delta) = d_1(p, p-\delta)$
- $\lambda_2(p,\delta) = d_2(p-\delta, p)$
- $\lambda_o(p,\delta) = d_1(p-\delta, p) + d_2(p, p-\delta) - d_1(p-\delta, p-\delta) - d_2(p-\delta, p-\delta)$.

Note that although we do not know the distribution over customer valuations, knowing the demand arrival rates $d_1(\cdot,\cdot)$ and $d_2(\cdot,\cdot)$ at just 3 price points is enough to estimate the arrival rates in the opaque selling model for a given p and δ. In practice, these demand functions may be readily available in many settings, particularly when retailers tend to experiment with pricing on a continual basis. Moreover, if δ is small, then these demand rates may be easily estimated using the price and cross-price elasticities of demand. We also note that Theorem 6 holds even if the valuations for product 1 and 2 are not symmetric. In the next subsection, we will consider an additive demand model and study the impact of opaque selling in this setting.

3.4.2 Profit, Revenue, and Costs under a Linear Demand Model

In this subsection, we will implement our results from the previous sections to see how opaque selling affects cost, revenue, and overall profit. In our experiments, we will first find the optimal price, p^*, for the traditional selling strategy. Then, we consider two opaque sells strategies, Opaque1 and Opaque2. Opaque1 will keep the prices of both transparent products at p^*, and then optimize over δ, which models the scenario where the retailer cannot change the actual price of the original products. Opaque2 will optimize over both p and δ, and thus is a dominant strategy over Opaque1. Our optimization procedure was to simply use a brute force line search over the entire variable space.

We will model the demand using a linear demand model for substitutable goods of the form $d_1(p_1, p_2) = a - bp_1 + dp_2$ and $d_2(p_1, p_2) = a - bp_2 + dp_1$. The ratio of d/b reflects the degree of substitutability, with a higher ratio implying a higher degree of substitutability. We will let $a = 10$, $b = 1$, and $d = 0.5$. (In Figure C-1 of Appendix C.3, we report similar results for the same setup except with $d = 0.1$.) For the supply chain parameters, we set the reorder level $c = 50$ and holding cost parameter $H = 1$. We test over different values of K until a positive profit is no longer achievable. In Figure 3-5, we report the profit, opaque levels, percent change in cost, and percent change in revenue under these parameters.

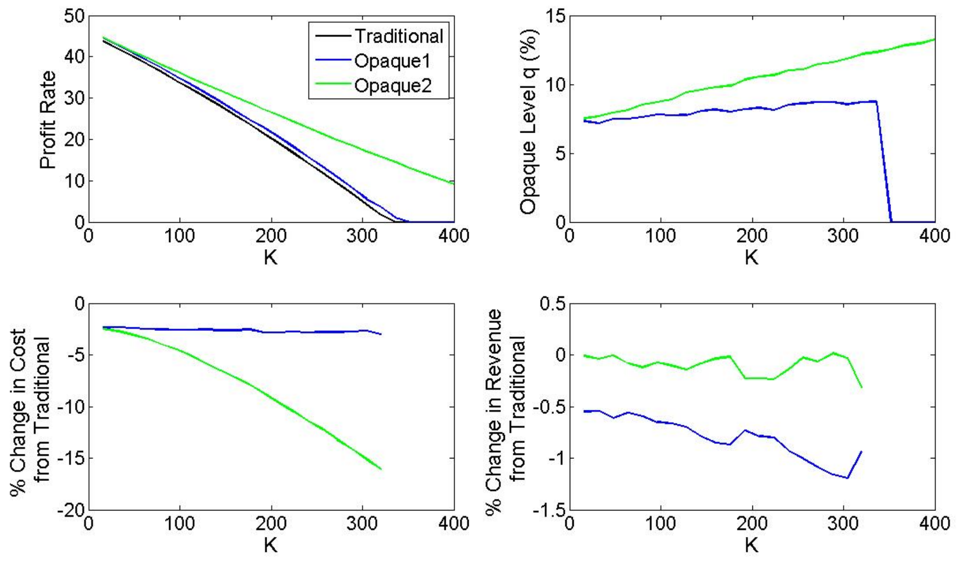

Figure 3-5: The black lines correspond to the optimal traditional strategy, the blue lines to the optimal Opaque1 strategy, and the green lines to the optimal Opaque2 strategy. The graphs correspond to the profits, opaque levels, percent change in cost, and percent change in revenue.

From Figure 3-5, there are several interesting phenomena that occur. First, the Opaque1 strategy tends to consistently provide a cost savings of about 2.5% while losing just less than 0.5% in revenue. The opaque level q remains roughly between 7-8%. The actual profit percent improvement is increasing in K since the profits of traditional selling are tending towards zero, while the Opaque1 profit remains above the traditional profit. In fact, there are regimes where the traditional strategy makes no profit while the opaque strategies are still profitable. Meanwhile the Opaque2 strategy provides a lot more value, due to its ability to retain almost the same amount of revenue, within 1.5%, but with less customers. The Opaque2 strategy is essentially using price differentiation to capture more revenue from picky customers, less customers overall, and a higher ratio of opaque customers. As the fixed ordering cost increases, the ability to price differentiate is crucial to capture the smaller potential set of customers. However, we should be aware of the fact that a linear demand model may not accurately reflect consumer price sensitivity as prices increase. Nevertheless, given appropriate demand models, a similar analysis can be performed by a retailer to determine the best form of opaque sales strategies and the corresponding potential improvements.

3.5 Comparison to a Dynamic Pricing Strategy

In this section, we will compare our opaque selling strategy to a particular class of dynamic pricing strategies. Most online retailers currently implement dynamic pricing strategies for reasons such as seasonality effects, competition, and inventory positioning. However, one major challenge of dynamic pricing is the difficulty to compute optimal policies, particularly in the presence of multiple products and inventory costs. To the best of our knowledge, there are no standard methodologies or analytical results for these scenarios. In addition, dynamic pricing strategies are often susceptible to strategic customers that may wait for their desired product to decrease in price before purchasing. In this section, we show that our opaque selling strategy, under some assumptions, can outperform a natural class of dynamic pricing policies for a traditional selling model. Although our theoretical result has limitations, we believe that it provides strong justification for an opaque selling strategy to be considered as a potential alternative to dynamic pricing.

Remember that we define an opaque sales strategy as selling two products at a price p, and an opaque product at price $p - \delta$. We will refer to this policy as OS(δ). Note that p is assumed to be fixed, although δ may vary over time. Since an optimal dynamic pricing is difficult to compute, we will instead of focus on the following natural class of dynamic pricing policies. At any point in time, one of the two products is sold at price p, and the other product is sold at a price $p-\delta$. We will refer to this policy as DP(δ). Again, note that p is assumed to be fixed, although δ may vary over time. This policy is motivated by that fact that many products have a standard price, and occasionally go on sale. The major restriction that we have placed is that at least one of the products is always at full price.

We will assume customers make decisions according to the same choice model as in Section 3.4, and the definitions of $d_1(p_1,p_2)$ and $d_2(p_1,p_2)$ are also the same. We also will define $d(p_1,p_2) = d_1(p_1,p_2) + d_2(p_1,p_2)$. In the next proposition, we show that if there are no picky and cheap ($d(p,p-\delta) = d(p-\delta,p-\delta)$) customers, then the profit of OS(δ) is greater than DP(δ). In fact, OS(δ) has a revenue and cost at least as good as DP(δ). Our assumption of no picky and cheap customers essentially means that once one product goes on a discount δ, all cheap customers will buy and no further customers are expected if the other product goes on a discount.

Proposition 7. *Assume that $d_1(\cdot,\cdot) = d_2(\cdot,\cdot)$ (symmetric demand) and $d(p,p-\delta) = d(p-\delta,p-\delta)$ (no picky and cheap customers). Then the cost, revenue, and profit rates of OS(δ) are at least as good as DP(δ). Moreover, the result is independent of the inventory dynamics and policy.*

Proof. Without loss of generality, assume that DP(δ) prices product 1 at p and product 2 at $p-\delta$. By definition, the demand rates for products 1 and 2 under DP(δ) is equal to $d_1(p,p-\delta)$ and $d_2(p,p-\delta)$. The total demand rate is just $d(p,p-\delta)$.

Now consider OS(δ), the demands for the products is provided by Theorem 6. Namely, the demand rate for product 1 is $d_1(p,p-\delta)$, $d_2(p-\delta,p)$ for product 2, and $d_1(p-\delta,p) + d_2(p,p-\delta) - d_1(p-\delta,p-\delta) - d_2(p-\delta,p-\delta)$ for the opaque product. The total demand rate is then $d(p,p-\delta) + d(p,p-\delta) - d(p-\delta,p-\delta)$, which is equivalent to $d(p,p-\delta)$ based on our assumptions. If a customer purchases an opaque product, we will fulfill the order using product 2 (not necessarily optimal strategy), and thus the effective demand rates for product 1 and 2 becomes exactly

$d_1(p, p-\delta)$ and $d_2(p, p-\delta)$.

We have now shown that DP(δ) and OS(δ) have the exact same effective demand rates for products 1 and 2. Therefore, one can easily see that the cost rate of these systems is exactly equal. In fact, using a coupling argument for customer arrivals, clearly the cost is the same regardless of the underlying inventory dynamics and policy being used.

Finally, we can easily that the revenue rate of DP(δ) is $pd(p, p-\delta) - \delta d_2(p, p-\delta)$ and the revenue rate of OS(δ) is $pd(p, p-\delta) - \delta(d_1(p-\delta, p) + d_2(p, p-\delta) - d_1(p-\delta, p-\delta) - d_2(p-\delta, p-\delta))$. Thus, the OS($\delta$) policy makes $\delta(d(p-\delta, p-\delta) - d_1(p-\delta, p))$ more revenue. This is directly due to the fact that only cheap customers get a discount in OS(δ), where as all customers of product 2 get a discount in DP(δ). The profit rate of OS(δ) is clearly better as well since it dominates on both revenue and cost. $\square$

This proposition implies opaque selling can outperform a particular class of dynamic pricing policies under a specific set of assumptions we have made. Specifically, if one were restricted to a dynamic pricing policy as described by DP(δ), where δ is chosen dynamically, then a strictly better policy would be to just use OS(δ), where the δ values are chosen dynamically exactly the same way. Therefore, the class of policies described by OS($\cdot$) dominate those of DP($\cdot$) under our assumptions. One other advantage of OS(δ) policy is that it less immune to strategic customers who have a strong preference. Specifically, in the DP policy, strategic customers can wait until their preferred item is discounted, where as in the OS policy this could never happen. Although a more thorough investigation is needed, we believe Theorem 7 provides strong justification to consider opaque selling as a potential alternative to dynamic pricing.

3.6 Extensions

In this section, we describe several extensions to the basic model described in Section 3.2. We first consider the case where the demand for the products is not identical. We then look at the scenario where there are more than two transparent products. Finally, we develop a generalization of the opaque product called the k-opaque product that allows the consumers to face less uncertainties. In all of these settings, we analyze the impact on the supply chain cost savings, and our results are consistent with those of the basic model in Section 3.3.

3.6.1 Asymmetric demand

Thus far we have assumed that the demand between the two products is identical. Naturally, it is of interest to consider the scenario when one product has more demand than the other. In this setting, the optimal fulfillment strategy for opaque products is no longer obvious, and we must rely on a dynamic program to make the optimal decisions. Let q_1 be the fraction of customers who prefer product 1 over product 2. With probability q, a customer will buy the opaque product. Furthermore, the reorder level for the two products may not be identical, and we shall refer to them as c_1 and c_2. In this setting, we can use a dynamic program to compute the expected number of customers that arrive between replenishments, and then use the same formulas from Theorem 2 to compute the supply chain costs.

For a given q_1 and q, let $J(x_1, x_2)$ denote the expected number of customers that arrive between replenishments, under an optimal fulfillment policy. If x_1 or x_2 is 0, then clearly $J(x_1, x_2) = 0$. Otherwise,

$$J(x_1, x_2) = 1+q_1(1-q)J(x_1-1, x_2)+(1-q_1)(1-q)J(x_1, x_2-1)+q\max(J(x_1-1, x_2), J(x_1, x_2-1)).$$

Using dynamic programming, we can compute all the values of J by using backwards induction on the total inventory remaining. In Figure 3-6, we look at 4 cases of asymmetric demand and compute the ordering cost savings. Specifically, we look at the cases where $q_1 = 0.6$ and $q_1 = 0.8$. The reorder levels are either $c_1 = c_2 = 50$ or $c_1 = 100q_1$ and $c_2 = 100(1 - q_1)$. Our choices for the reorder levels are motivated by the scenarios where the supplier wants to have equal reorder levels or wants them to be in proportion to the imbalance in demand.

There are several interesting observations about Figure 3-6. First, the value of

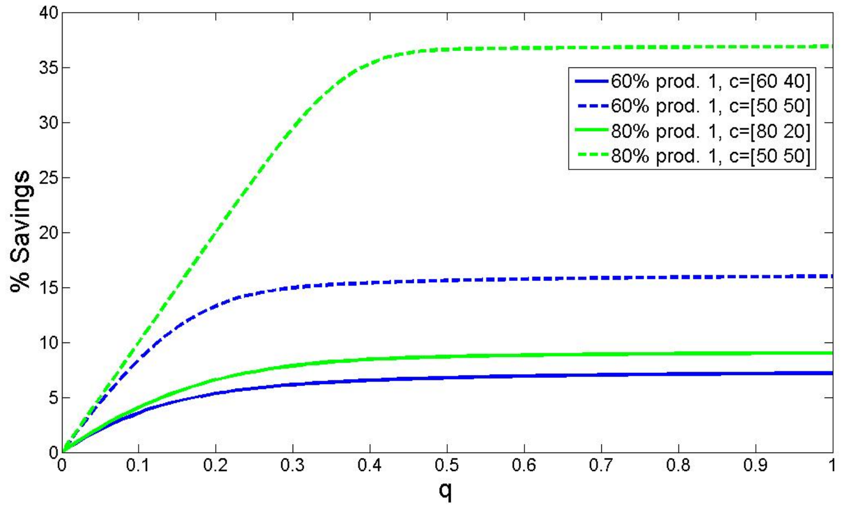

Figure 3-6: We compute the ordering cost savings achieved when the demand is imbalanced between the two products. The blue lines represent the scenario where 60% of customers prefer product 1, and the green line represents the scenario where 80% of customers prefer product 1. The solid lines are for the cases where the reorder levels are proportional to the arrival rates, and the dashed lines are for the case where the reorder levels are identical between the two products.

opaque selling gets larger as q_1 gets larger, independent of the reorder levels. This seems to suggest that the more imbalanced the demand may be, even if it is unknown a priori, the more crucial opaque products will be for reducing supply chain costs. Second, we can see that if the reorder levels are the same for products 1 and 2, this results in a large imbalance in the system and thus opaque selling has more value in this setting. When the reorder levels are proportional to the arrival rates, we observe a behavior similar to the symmetric demand case, with savings on a similar order of magnitude. Overall, this graph strongly implies that opaque products have a similar, if not greater, impact when demand is imbalanced.

3.6.2 N transparent products

The model we proposed in this work can be easily extended to N transparent products, for any integer $N > 2$. Under the setting where the retailer is selling N transparent products with identical demand and one opaque product, it is simple to check that it is still optimal to fulfill an opaque product with the product of the highest on-hand inventory at the time. Moreover, it is easy to check that similar to Theorem 2, both the ordering and holding cost savings can be computed by evaluating $\mathbb{E}[R(q, c)]$, the expected number of customers served between consecutive replenishments when the fraction of opaque customers is q and the reorder level is c. Therefore, given any integers c and $N > 2$, we can numerically compute the cost savings of having q fraction of customers purchasing the opaque product.

In Figure 3-7, we consider the scenario where there are N products with identical demand and evaluate the cost savings achieved for different values of c and N. We focus only on the ordering cost savings, as the holding cost savings have similar characteristics. From Figure 3-7, we observe that the value of opaque product selling increases significantly as the number of products gets larger. Note that we focus on the scenario when the opaque level is 10%, i.e. $q = 0.1$, as this is a relatively practical scenario. When there are 10 products, which is not unusual when products vary by color, the savings can be up to 7% which is quite significant.

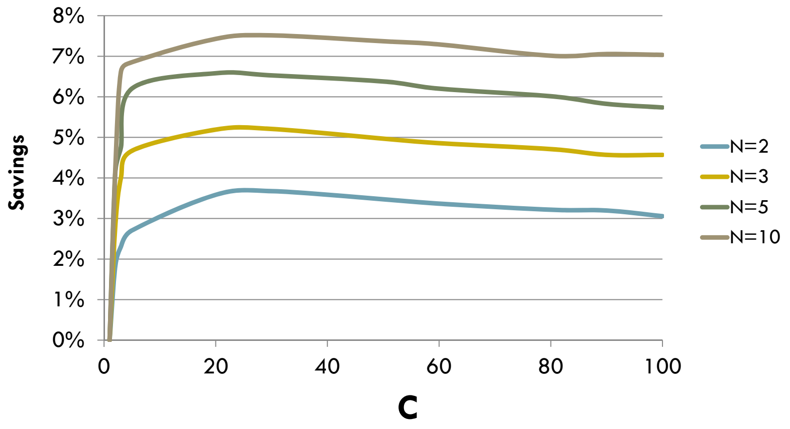

Figure 3-7: We compute the ordering cost savings achieved for different values of N and c when $q = 0.1$.

3.6.3 k-opaque products

We now introduce a generalization of opaque products called a k-opaque product. If a customer purchases a k-opaque product, then the customer can first choose k out of the N products they are willing to receive, and then the retailer provides them with one of the k chosen products. Therefore, a 1-opaque product is equivalent to no opaque selling, and an N-opaque product is equivalent to a regular opaque product. The choice of k allows the supplier to give more choice to the customer (and therefore a smaller discount), at the expense of having less flexibility for allocating opaque products.

In Figure 3-8, we evaluate the impact of k-opaque products when $N = 5$ and the opaque level is $q = 0.1$ for two different reorder levels corresponding to $c = 20$ and $c = 100$. For simplicity, we assume that all subsets of size k are equally likely to be chosen. Observe that the savings of 2-opaque products and N-opaque products is only different by about 1%. This implies that if we allow customers to have more choice, then we can still capture most of the potential savings compared to the scenario where customers get any one of the N products at random. By offering 2-opaque products to customers, this means we can reduce δ without sacrificing much of cost savings. Ideally, a retailer could optimize over p, δ, and k when implementing an opaque sales strategy.

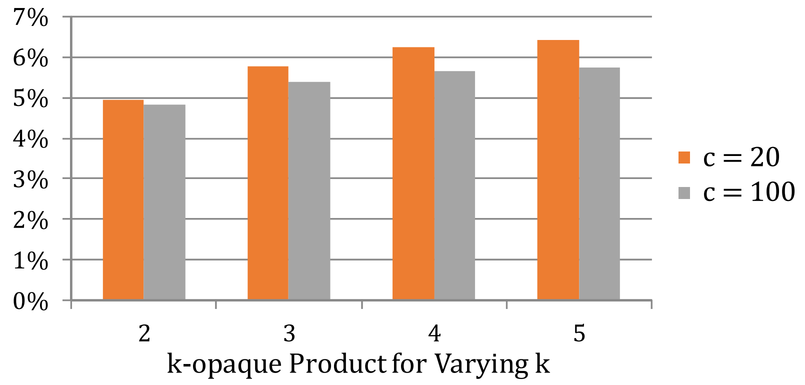

Figure 3-8: We compute the ordering cost savings when using k-opaque products for different values of k and two reorder levels 20 and 100.

3.6.4 Future Directions

We believe this work has provided strong evidence and tools for using opaque products in an online retail environment. Consequently, there are a variety of potential research directions to consider. For example, one may need to consider alternative inventory policies than the ones used here, as well as how to compute optimal inventory policies with opaque products. In addition, it would be of value to consider alternative ways to estimate q, and even do real-life experimentation to observe the behavior of customers with opaque products. Finally, it would interesting to generalize our theoretical results when demand is asymmetric or there are more than two products.

Chapter 4

The Submodular Joint Replenishment Problem

4.1 Introduction

Inventory models with deterministic and non-stationary demand have a long and rich history in supply chain management, beginning with the seminal paper of Wagner and Whitin [89]. One of the main features of these models is to capture the tradeoff between holding costs and fixed setup costs. Setup costs typically represent, among others, the use of machines, trucks, and/or laborers. When multiple item types are ordered in the same period, some of the fixed costs are typically shared among the different item types. In most inventory management models, the economies of scale are traditionally captured by a *joint setup cost* structure, and the problem is generally known as a *joint replenishment problem (JRP)*. The JRP has been used in many applications including inventory management, maintenance scheduling, logistics, and transportation problems.

The most traditional JRP model uses the *additive* joint setup cost structure. In this model, there is a joint setup cost if *any* item type is ordered, in addition to an individual item setup cost for each item type ordered. The additive joint setup cost structure is clearly very limited in some cases, yet there is a surprising lack of results for models with more complex joint setup cost structures when demand is non-stationary. This in stark contrast to Queyranne [72], Federgruen and Zheng [31], and Teo and Bertsimas [84] which gave near-optimal algorithms for inventory models with a very general setup cost structure but with *constant* demand. In this work, we consider several generalizations of the deterministic, *non-stationary* joint replenishment problem beyond the additive cost structure that capture many interesting settings in practice. Since joint replenishment problems with non-stationary demand are typically NP-hard, even for the additive model (Arkin et al. [1]), it is computationally intractable to find optimal solutions quickly. However, our algorithms can efficiently find solutions that are provably close to optimal, and in the worst case are guaranteed to be within a fixed constant factor of the optimal cost.

In the models studied in this work, there are multiple item types, each with a sequence of demands over a discrete time horizon of finitely many periods. That is, there is a specified demand quantity for each item type due in each time period. Each demand must be completely satisfied by an order at or prior to its due date period, which means neither backlogging nor lost sales are allowed. The cost of satisfying the demand is composed of fixed setup costs and holding costs. The joint setup cost for ordering in any time period is a function of the specific subset of item types ordered. The setup cost function we study satisfies two natural properties known as *monotonicity* and *submodularity*. The monotonicity property simply means that as more item types are ordered, the total setup cost increases. The submodularity property captures the economies of scale in ordering more item types, i.e., the marginal cost of adding any specific item type to a given order decreases as the given order increases. The holding cost for each demand point depends on the item type and the length of time the inventory was held. We assume no capacity constraints and stationary per unit variable costs. One can easily show that zero-inventory ordering policies are optimal in this model, and thus every demand of a given item type is satisfied by the latest order of that item prior to its due date period. The goal is to satisfy all the demands on time by a sequence of orders that minimizes the total setup costs plus holding costs. We refer to this problem as the *submodular joint replenishment problem.*

For the submodular JRP, we show that a simple greedy algorithm achieves an approximation ratio that is logarithmic in the total number of demands. In addition, we also describe an integer programming formulation whose linear programming relaxation can be solved efficiently and has special structure for the optimal solution. We then consider three special cases of the submodular JRP with non-stationary demand that capture a wide range of applications. These cases are called the *tree JRP*, *laminar JRP*, and the *cardinality JRP*, none of which have any clear mathematical relation with the other.

In the *tree JRP* case, we are given a rooted tree where every node represents a process that incurs a setup cost, and the leaves of the tree represent the item types. The joint setup cost of ordering any subset of item types is the cost of all the nodes on the paths from the root to the leaves corresponding to the item types being ordered. The tree JRP model captures situations where each item type requires a chain of processes to be performed, and several of those processes are shared by other item types. One application of the tree JRP is in maintenance scheduling problems (Levi et al. [59]) for aircraft engines. Each module of the engine corresponds to a node in the tree, and to get to a certain engine part requires removing all necessary modules. The tree with only a root and leaves is identical to the additive joint setup cost structure, and thus this problem is also NP-hard. We provide an approximation algorithm that is no more than three times the optimal offline cost. This algorithm is similar in spirit to Levi et al. [58], which only considered the additive JRP. Specifically, the algorithm is based on solving a linear program, and then successively rounding the variables corresponding to the nodes in the tree in a particular fashion. Specifically, we start by opening orders containing the root process, and work our way down the tree using a breadth-first search. For each node, we round the corresponding variables to a time where the parent node has been ordered, thus ensuring that all necessary processes for an item type are accounted for when it is ordered.

In the *laminar JRP* case, the submodular cost function is specified through a laminar family of subsets of item types, each with an associated cost. The family can be modeled as a tree where the nodes are the subsets, and the children of each node must partition the corresponding subset of item types belonging to that node. This captures situations where there are machines (or laborers) with varying degrees of specialization. Any two machines have the property that either they cannot make any of the same item types, or that one machine has strictly more capabilities than the other. The joint setup cost for ordering a set of item types is simply the setup cost of the cheapest machine that is capable of producing all the item types being ordered. In other words, the cost of ordering a subset of items is equal to the cost associated with the node in the tree corresponding to the minimal subset that includes the ordered items. For the laminar JRP, we are surprisingly able to solve the problem to optimality with an efficient dynamic program, which means that this variant is not NP-hard. The subproblems of the dynamic program are finding the optimal cost of serving the demands in a specific interval, given that a specific machine will be in use at the beginning of the interval. The proof of correctness relies on our ability to decompose the item types into groups based on which machines they can be processed on.

Finally, the *cardinality JRP* is the case where the joint setup cost function has the property that the cost of ordering a subset of items types is simply a function of the cardinality of the subset being ordered. The submodularity in this case implies that the setup cost is concave in the cardinality of the set of item types being ordered. A natural application of this model is when all the item types are very similar, but vary in only one aspect such as color or size. Although the cardinality JRP is NP-hard, which was shown indirectly in Arkin et al. [1], we provide an efficient algorithm with a worst-case approximation ratio of five. This algorithm is based on an innovative iterative rounding procedure that uses the variables from a linear relaxation of a novel integer programming formulation of the cardinality JRP. Our algorithm carefully builds up orders based on their size, while ensuring that the cost of any particular order can be paid for using the primal objective costs. The holding costs are accounted for using a property of the respective dual linear program.

4.1.1 Literature Review

Joint replenishment problems are infamous for being intractable, and thus have been typically studied via the notion of *approximation algorithms*. An approximation algorithm is an algorithm that finds a feasible solution efficiently, and guarantees that its cost is always within a certain factor of the respective optimal cost. This factor is called the *approximation ratio*. When demand is assumed to be stationary and continuous for the additive JRP, Schulz and Telha [76] showed that this problem is as hard as integer factorization. For this problem, Roundy [75] showed that "power-of-two" policies have approximation ratios of 1.06 and 1.02, depending on whether the base planning period is fixed or not. In Teo and Bertsimas [84], an improved approximation ratio of 1.04 was obtained for the additive JRP with fixed base planning periods. When the time horizon is finite, Segev [77] and Nonner and Sviridenko [67] provide quasi-polynomial and efficient polynomial time approximation schemes, respectively. In Federgruen and Zheng [31], the results of Roundy [75] were generalized for the submodular JRP with constant demand. Other stationary inventory models with submodular costs were considered in Federgruen et al. [33], Herer and Roundy [44], and Viswanathan [88].

The literature for the submodular JRP with non-stationary demand has focused primarily on the additive joint setup cost structure, which was shown to be NP-hard in Arkin et al. [1]. Nonner and Souza [66] further showed that this problem is APX-hard when holding costs are nonlinear with respect to time, which is the case for the models we consider. Several heuristics for the non-stationary additive JRP have been proposed with varying degrees of theoretical performance guarantees in Veinott Jr [87], Zangwill [95], Kao [52], Joneja [51], Federgruen and Tzur [30], Levi et al. [57], and Stauffer et al. [82]. The current best approximation algorithms for the additive JRP with non-stationary demand are due to Levi et al. [58], which has an approximation ratio of 1.80, and Bienkowski et al. [10] which has an approximation ratio of 1.791. Becchetti et al. [6] and Khanna et al. [54] have considered special cases of the tree JRP with assumptions on the holding cost structure. Chan et al. [19] show that zero-inventory policies are near-optimal for joint replenishment problems with piecewise linear costs, however their conditions do not imply submodularity.

In Section 4.2, we give a precise mathematical description of the submodular JRP along with a greedy approximation algorithm and an integer programming formulation. We also explore some properties regarding the linear programming relaxation. We then consider the tree, laminar, and cardinality JRPs in Sections 4.3, 4.4, and 4.5. Finally, we offer some concluding remarks and possible research directions in Section 4.6.

4.2 Submodular JRP

4.2.1 Model

In this section, we describe the submodular joint replenishment problem. There are N item types in total denoted by the set $\mathcal{N} := \{1, \dots, N\}$. For each item type $i \in \mathcal{N}$, there are demands that need to be satisfied over a planning horizon of T periods. The demand for each item type $i \in \mathcal{N}$ and time period $t = 1, \dots, T$ is denoted by $d_{it} \geq 0$. Each demand point (i, t) with positive demand needs to be served by an order of item type i placed no later than t. The two components to the total cost incurred are the fixed ordering cost and holding cost.

We let $K(S)$ denote the joint setup cost of ordering the set of item types in S in any given period. $K(\cdot)$ is assumed to be nonnegative, non-decreasing and submodular. The non-decreasing assumption means that for every $S_1 \subseteq S_2 \subseteq \mathcal{N}$, $K(S_1) \leq K(S_2)$. The submodularity assumption means that for every set $S_1, S_2 \subseteq \mathcal{N}$, $K(S_1)+K(S_2) \geq K(S_1 \cup S_2)+K(S_1 \cap S_2)$. This definition can be shown to be equivalent to the following: for every set $S_1 \subseteq S_2 \subseteq \mathcal{N}$ and any item type $i \in \mathcal{N}$, $K(S_1 \cup \{i\}) - K(S_1) \geq K(S_2 \cup \{i\}) - K(S_2)$. This alternative definition more clearly conveys the economies of scale interpretation of submodularity, i.e., that the marginal cost of adding an item type to a given order is decreasing as more item types are included in the given order. In the traditional additive JRP, the joint setup cost function is defined as

$K(S) := K_0 + \sum_{i \in S} K_i$. It is easy to see that this satisfies the monotonicity and submodularity properties, hence the additive JRP is a special case of the submodular JRP.

We let h^i_{st} be the per unit holding cost of item i from period s to t. We assume h^i_{st} is nonnegative and non-increasing in s for a fixed i and t. For convenience, we also denote the cost of holding inventory for demand (i,t) from an order in period s as $H^i_{st} := d_{it} h^i_{st}$.

In Theorem 14, we provide an $O(\log(NT))$-approximation algorithm to the submodular JRP. The proof relies on framing it as an example of the set cover problem, which is similar in spirit to a reduction in Svitkina and Tardos [83] for a facility location type problem. [78] have recently been able to show a $O(\log N)$-approximation algorithm for the same problem.

Theorem 14. *There exists an $O(\log(NT))$-approximation algorithm for the submodular JRP by using a greedy algorithm for the set cover problem.*

Proof. We will reduce the submodular JRP to the set cover problem. In the set cover problem, there are m objects that need to be covered, and n subsets of those objects that each have a different cost. The goal is to cover the m objects using the cheapest collection of available subsets. Chvatal [24] showed that a simple greedy algorithm is an $O(\log m)$-approximation algorithm to this problem. The algorithm works by iteratively adding the subset who has the smallest ratio of its cost over the number of uncovered objects in the subset.

To model the submodular JRP as a set cover problem, we let each demand point be an object, which means there are NT in total. Let $\mathcal{D}$ be the set of all demand points. Then the possible subsets we may use in the cover is denoted by $\mathcal{U} := \{1, \dots, T\} \times 2^{\mathcal{D}}$. The cost of a subset $(t, U) \in \mathcal{U}$, denoted by $c(t, U)$, is simply the cost of serving the demand points in U from time t in the submodular JRP. More specifically, $c(t, U) = \infty$ if U contains a demand point with due date before t. Otherwise, if we let $I(U)$ denote the subset of item types in U, then $c(t, U) = K(I(U)) + \sum_{(i,u) \in U} H^i_{tu}$. Notice that $c(t, U)$ is a submodular function in U for a fixed t due to the submodularity of $K(\cdot)$

and the fact that the holding costs are separable.

Now we need to show that we can find a set $(t, U) \in \mathcal{U}$ whose ratio of $c(t, U)$ over the number of uncovered demand points is smallest. Since the number of such possible sets is exponential, we cannot enumerate and thus need to define an efficient procedure. To find the set to add to the cover, we first find the set who has the smallest ratio for each time period t, and then choose the cheapest among them. Let $w_{(i,t)} = 1$ if demand point (i, t) is still uncovered and let it be 0 otherwise. For a specific time period t, we aim to find a set $U \in 2^{\mathcal{D}}$ that minimizes $\frac{c(t,U)}{\sum_{(i,t)\in U} w_{(i,t)}}$. This problem is equivalent to finding the minimum α, via binary search, for which there exists a set $U \in 2^{\mathcal{D}}$ such that $\frac{c(t,U)}{\sum_{(i,t)\in U} w_{(i,t)}} \leq \alpha$. Equivalently, we can write this inequality as $c(t, U) - \alpha \sum_{(i,t)\in U} w_{(i,t)} \leq 0$. Since the left hand side of this expression is clearly submodular, we can find a set U that satisfies this inequality for a given α by solving a submodular minimization problem, which is known to be efficiently solvable ([61]). Therefore, we have now proved the main theorem. $\square$

4.2.2 Integer Programming Formulation

The following is an integer programming formulation of the submodular JRP. The binary variable y_s^S is 1 if the subset of item types S is ordered in period s and is 0 otherwise. The binary variable x_{st}^i is 1 if demand (i, t) is satisfied using an order from period s and is 0 otherwise.

$$\begin{aligned}
\text{minimize} \quad & \sum_{S\subseteq\mathcal{N}} \sum_{s=1}^{T} K(S) y_s^S + \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{s=1}^{t} H_{st}^i x_{st}^i && \text{(IP)} \\
\text{subject to} \quad & \sum_{s=1}^{t} x_{st}^i = 1, \qquad i = 1, \ldots, N, t = 1, \ldots, T && (4.1) \\
& x_{st}^i \leq \sum_{S:i\in S\subseteq\mathcal{N}} y_s^S, \quad i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots t && (4.2) \\
& x_{st}^i, y_s^S \in \{0, 1\}, \quad i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots t, S \subseteq \mathcal{N}
\end{aligned}$$

We first argue that this is indeed a valid formulation for the submodular JRP.

Constraint (4.1) ensures that each demand (i, t) is completely served by an order at some time $s \leq t$. Constraint (4.2) ensures that if any demand (i, t) is served by an order at time s, then there must be a subset ordered at time s that includes i. By submodularity, the optimal solution will only set at most one y variable to 1 in any time period s. Next we get the natural LP relaxation of (IP) by relaxing the integer constraints on the variables.

$$
\begin{aligned}
\text{minimize} \quad & \sum_{S \subseteq \mathcal{N}} \sum_{s=1}^{T} K(S) y_s^S + \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{s=1}^{t} H_{st}^i x_{st}^i && \text{(P)} \\
\text{subject to} \quad & \sum_{s=1}^{t} x_{st}^i = 1, \qquad i = 1, \ldots, N, t = 1, \ldots, T && (4.3) \\
& x_{st}^i \leq \sum_{S: i \in S \subseteq \mathcal{N}} y_s^S, \; i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots t && (4.4) \\
& x_{st}^i, y_s^S \geq 0, \qquad i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots t, S \subseteq \mathcal{N}
\end{aligned}
$$

The following lemma shows that there exists an optimal solution to (P), such that in any given time period, the set of y variables that are positive have a very special structure. This lemma will be used later to provide better formulations for the special cases of the submodular JRP that we consider.

Lemma 23. *There exists an optimal solution (x, y) to (P) such that for any given period t and any two subsets $R \subseteq \mathcal{N}$ and $S \subseteq \mathcal{N}$, if $y_t^R > 0$ and $y_t^S > 0$ then they are nested, i.e., either $R \subseteq S$ or $S \subseteq R$.*

Proof. Given an arbitrary optimal solution $(\hat{x}, \hat{y})$ to (P), we will construct an another optimal solution (x, y) with the desired property. Fix a time period t. Let $Z_t^i := \sum_{S: i \in S} \hat{y}_t^S$ be the sum of the fractional number of sets ordered in period t that contains item type i. Now consider the following auxiliary optimization problem which has a variable r_t^S for every set $S \subseteq \mathcal{N}$.

$$\begin{aligned} &\text{minimize} \quad \sum_{S\subseteq\mathcal{N}} K(S) r_t^S && \text{(AUX-t)}\\ &\text{subject to} \quad \sum_{S:i\in S} r_t^S \geq Z_t^i, \qquad i=1,\ldots,N && (4.5)\\ &\qquad\qquad\quad r_t^S \geq 0 \qquad\qquad S\subseteq\mathcal{N} \end{aligned}$$

Recall $K(S)$ is the cost of ordering the item types in S and thus (AUX-t) exactly captures the cheapest way for ordering items so that each item i is ordered at least Z_t^i. Since $K(S)$ is a submodular function, then (AUX-t) is the dual of polymatroid where the greedy algorithm finds an optimal solution (see Bertsimas and Weismantel [7]). Specifically, assume without loss of generality that the items are indexed such that $Z_t^1 \geq Z_t^2 \geq \ldots \geq Z_t^N$ and let $S(i) := \{1,\ldots,i\}$. Then an optimal solution to (AUX-t) is given by $r_t^{S(N)} = Z_t^N$, and $r_t^{S(i)} = Z_t^i - Z_t^{i+1}$ for $i = 1,\ldots,N-1$, and $r_t^S = 0$ for all other S. Note that by construction $S(i)$ has the desired nested property. Thus, setting $x = \hat{x}$ and $y_t^S = r_t^S$, where r_t^S is the optimal solution to (AUX-t), gives a feasible solution to (P) where every demand is served in exactly the same manner. Thus, the holding costs are identical for the two solutions, and the setup costs of (x,y) are no more than that of $(\hat{x},\hat{y})$ since each y_t^S gives a feasible solution to (AUX-t). Therefore, (x,y) is also an optimal solution, and in addition satisfies the desired nested property. $\square$

Now we consider the dual program of (P). Let b_t^i and l_{st}^i be the dual variable corresponding to first and second set of constraints in (P), respectively, then the dual program of (P) is:

$$\text{maximize} \quad \sum_{i=1}^{N}\sum_{t=1}^{T} b_t^i \tag{D}$$

$$\text{subject to} \quad b_t^i \leq H_{st}^i + l_{st}^i, \qquad i = 1,\dots,N, t = 1,\dots,T, s = 1,\dots,t \tag{4.6}$$

$$\sum_{i\in S}\sum_{t=s}^{T} l_{st}^i \leq K(S), \quad s = 1,\dots,T, S \subseteq \{1,\dots,N\} \tag{4.7}$$

$$l_{st}^i \geq 0 \qquad i = 1,\dots,N, t = 1,\dots,T, s = 1,\dots,t$$

Note that (P) contains an exponential number of variables and (D) contains an exponential number of constraints. One can solve the dual (and therefore the primal) using the ellipsoid method, since there is an efficient separation oracle by using submodular function minimization to find violated constraints. However, we focus on special cases of the submodular JRP that have polynomial-size LP relaxations. This also allows us to give very efficient LP-based approximation algorithms for these cases.

4.3 Tree Joint Replenishment Problem

In this section, we consider the tree JRP, where the joint setup cost structure is represented by a tree $\mathcal{T}$ with a root r. Specifically, there is a fixed cost K_j associated for each node j in the tree, and each item type i corresponds to a leaf in the tree. For every node $j \in \mathcal{T}$, let $path(j)$ denote the unique path from node j to the root of the tree. The cost of ordering a subset of item types $S \in \mathcal{N}$ is then equal to $\sum_{j\in\cup_{i\in S} path(i)} K_j$, i.e., the cost of all nodes that belong to the paths from r to the leaves corresponding to the set S. (See Figure 4-1 for an example.)

Note that this captures the additive JRP as a special case since the additive joint setup cost structure can be viewed as a tree. Specifically, there is a root node connected to N leaves, one for each item. The cost of the root node is K_0, the joint setup cost, and the cost of each leaf i corresponds to the item setup cost K_i. In the remainder of this section, we advance the ideas of Levi et al. [58] to give a

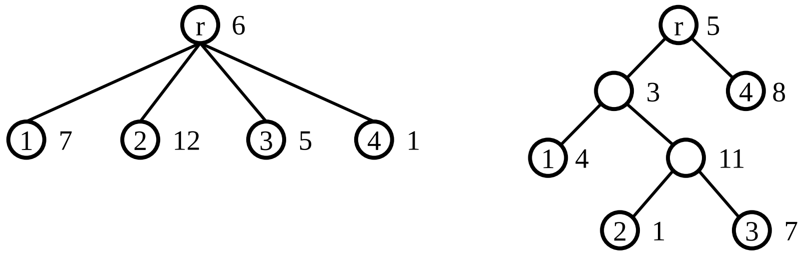

Figure 4-1: These two trees represent two examples of the tree joint setup cost structure. The leaves are labeled according to the item type they represent and the root node is labeled with an r. The number next to each node denotes the cost of that node. The left tree is the special case of an additive joint setup cost structure. In the right tree, the cost of ordering item types 1 and 3 is 5+3+4+11+7 = 30.

3-approximation algorithm for the tree-JRP via LP rounding.

The following is the natural LP relaxation of an IP formulation for the tree JRP. We use the binary variable y_s^j to indicate if the node j is ordered in period s, and binary variable x_{st}^i to indicate whether the demand d_{it} was served from period s.

$$\text{minimize} \quad \sum_{j\in\mathcal{T}}\sum_{s=1}^{T} K_j y_s^j + \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{s=1}^{t} H_{st}^i x_{st}^i \tag{P-T}$$

$$\text{subject to} \sum_{s=1}^{t} x_{st}^i = 1, \quad i = 1,\ldots,N, t = 1,\ldots,T \tag{4.8}$$

$$x_{st}^i \le y_s^j, \qquad i = 1,\ldots,N, t = 1,\ldots,T, s = 1,\ldots t, j \in path(i) \tag{4.9}$$

$$x_{st}^i, y_s^j \ge 0, \quad i = 1,\ldots,N, t = 1,\ldots,T, s = 1,\ldots t, j \in \mathcal{T}$$

The correctness of (P-T), and in particular for constraint (4.9) follows from the fact that in order to place an order for item i, one has to pay $\sum_{j\in path(i)} K_j$. Next, we show that (P-T) is just as strong as (P), the LP relaxation for the submodular JRP. We denote Z_{PT} and Z_P as the optimal values of (P-T) and (P), respectively.

Lemma 24. *For the tree JRP, (P-T) is equivalent to (P), i.e., $Z_{PT} = Z_P$.*

Proof. First we show that we can convert an optimal solution (x, y) of (P-T) to a feasible solution $(\hat{x}, \hat{y})$ of (P). We start by observing some properties about the optimal solution of (P-T).

1. For each node j in the tree that is not a leaf, let $C(j)$ be the set of children of j. Then for each time period s, we know $y_s^j = \max_{k \in C(j)} y_s^k$ or otherwise we can decrease y_s^j without affecting feasibility.

2. Consider any nodes j and j' such that $j' \in path(j)$. Then for each time period s, we know that $y_s^j \leq y_s^{j'}$. This follows directly from the previous fact.

3. For any time period s and any node i that is a leaf in the tree (i.e., i corresponds to an actual item type), we know $y_s^i = \max_t x_{st}^i$ since otherwise we can decrease the value of y_s^i without affecting feasibility.

Given the properties above, we can determine the set of orders that are induced by the optimal solution (x, y). In a particular time period s, let $\sigma(\cdot)$ be an ordering of the items $\{1, \ldots, N\}$ such that $y_s^{\sigma(1)} \geq y_s^{\sigma(2)} \geq \ldots \geq y_s^{\sigma(N)}$. Then the sets that are ordered in period s are of the form $S(i), i = 1, \ldots, N$, where $S(i) = \{\sigma(1), \sigma(2), \ldots, \sigma(i)\}$. Notice that as in Lemma 23, the sets ordered here are nested, i.e., $S(1) \subset S(2) \subset \ldots \subset S(N)$.

Now we are ready to describe the conversion of an optimal solution (x, y) of (P-T) to a feasible solution $(\hat{x}, \hat{y})$ of (P). Let $\hat{x} = x$ and let $\hat{y}_s^S = 0$ except for $\hat{y}_s^{S(N)} = y_s^{\sigma(N)}$, $\hat{y}_s^{S(i)} = y_s^{\sigma(i)} - y_s^{\sigma(i+1)}$ for $i = 1, \ldots, N-1$. Notice by construction the item type $\sigma(i)$ is in set $S(i), \ldots, S(N)$ and using a telescoping sum it is easy to verify that $\sum_{S:\sigma(i) \in S} \hat{y}_s^S = y_s^{\sigma(i)}$. Hence constraint (4.4) in (P) is satisfied due to constraint (4.9) of (P-T). The proof is completed by noticing that the cost of $(\hat{x}, \hat{y})$ in (P) is the same as (x, y) in (P-T) by construction.

For the converse, we describe how to convert an optimal solution $(\hat{x}, \hat{y})$ of (P) to a feasible solution (x, y) of (P-T). To construct a feasible solution for (P-T), we first let $x = \hat{x}$. By Lemma 23, for every time period s we may assume that the subsets S where $\hat{y}_s^S > 0$ are nested. Without loss of generality, we index the item types so that all sets ordered at time s are of the form $S(i) := \{1, \ldots, i\}$ for $i = 1, \ldots, N$. Now for each leaf node i (which corresponds to an item type), we let $y_s^i = \sum_{k=i}^{N} \hat{y}_s^{S(k)}$, which are all the subsets ordered in (P) that include item i. Now, for each node j that is not a leaf, let $C(j)$ be the set of children of j. Then set $y^j_s = \max_{k \in C(j)} y^k_s$. It is easy to check (x, y) is feasible for (P-T) and that the corresponding cost is the same as the cost of $(\hat{x}, \hat{y})$ for (P). This completes the proof since we have demonstrated that there is a one-to-one mapping between (P) and (P-T). □

Next we describe the dual of (P-T), which will be used in the analysis of our LP rounding algorithm. Let b^i_t and l^{ij}_{st} be the dual variables corresponding to constraints (4.8) and (4.9) in (P-T) respectively. The dual of the (P-T) is then:

$$\begin{aligned}
&\text{maximize} && \sum_{i=1}^{N} \sum_{t=1}^{T} b^i_t && \text{(D-T)}\\
&\text{subject to} && b^i_t \leq H^i_{st} + \sum_{j \in path(i)} l^{ij}_{st},\ i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots, t && (4.10)\\
& && \sum_{i: j \in path(i)} \sum_{t=s}^{T} l^{ij}_{st} \leq K_j,\ s = 1, \ldots, T, j \in \mathcal{T} && (4.11)\\
& && l^{ij}_{st} \geq 0, \qquad i = 1, \ldots, N, t = 1, \ldots, T, s = 1, \ldots, t, j \in \mathcal{T}
\end{aligned}$$

4.3.1 LP Rounding Algorithm

We will show how to round the optimal solution to (P-T), denoted by (x, y) to a feasible solution to the tree JRP problem with cost at most 3 times the optimal value of (P-T), thus obtaining a 3-approximation algorithm.

Our rounding procedure considers each node in the rooted tree one at a time, starting at the root node. The nodes can be processed in any order, as long as node j is processed last among all nodes in $path(j)$ (i.e., the nodes on the path from j to the root). Hence one can use Breadth First Search (BFS) or Depth First Search (DFS) starting from the root node.

We first describe the processing of the root node, where we decide in which time period to place orders. The rounding procedure is based on the values of $y^0_1, \ldots, y^0_T$, which are the variables corresponding to fractional orders of the root node in (P-

T). We place an order of the root node in period s if $(\sum_{t=1}^{s-1} y_t^0, \sum_{t=1}^{s} y_t^0]$ contains an integer. Now for each node j, assume we have already processed all other nodes in $path(j)$, including the parent node j'. We place a *tentative order* in period s for node j if $(\sum_{t=1}^{s-1} y_t^j, \sum_{t=1}^{s} y_t^j]$ contains an integer. Motivated by the fact that we can only order j if j' has been ordered, we place actual orders of j via a two-sided push procedure as follows. If there is a tentative order of j at period s, then place an actual order of j at the first order point of j' in $(s, T]$ and place another actual order of j at the latest order point of $j' \in [1, s]$ (if such orders of j' exist). Notice that by construction, the orders are synchronized. See Figure 4-2 for an example of the rounding procedure. The pseudo-code for the algorithm is given below.

Algorithm 1: LP Rounding Algorithm for the tree JRP

```
Solve (P-T)
Generate order points for the root node
// Process the rest of the nodes in BFS or DFS order
for each node $j$ do
    Generate tentative order points for node $j$
    for each time period $s$ that contains a tentative order of $j$ do
        if $\exists$ an order of $parent(j)$ in $(s, T]$ then
            Place an order for node $j$ at the earliest order point of $parent(j)$ in $(s, T]$
        if $\exists$ an order of $parent(j)$ in $[1, s]$ then
            Place an order for node $j$ at the latest order point of $parent(j)$ in $[1, s]$
Serve each demand point $(i, t)$ from the latest order up to time $t$ that includes item type $i$
```

4.3.2 Analysis

We first prove a structural lemma on the algorithm which is key to the subsequent performance analysis. We will refer to (x, y) as the optimal solution to $(P - T)$ in the following analysis.

Lemma 25. *For each node j and time interval of periods $[s, t]$ where $\sum_{r=s}^{t} y_r^j \geq 1$, the algorithm places an order for node j somewhere in $[s, t]$.*

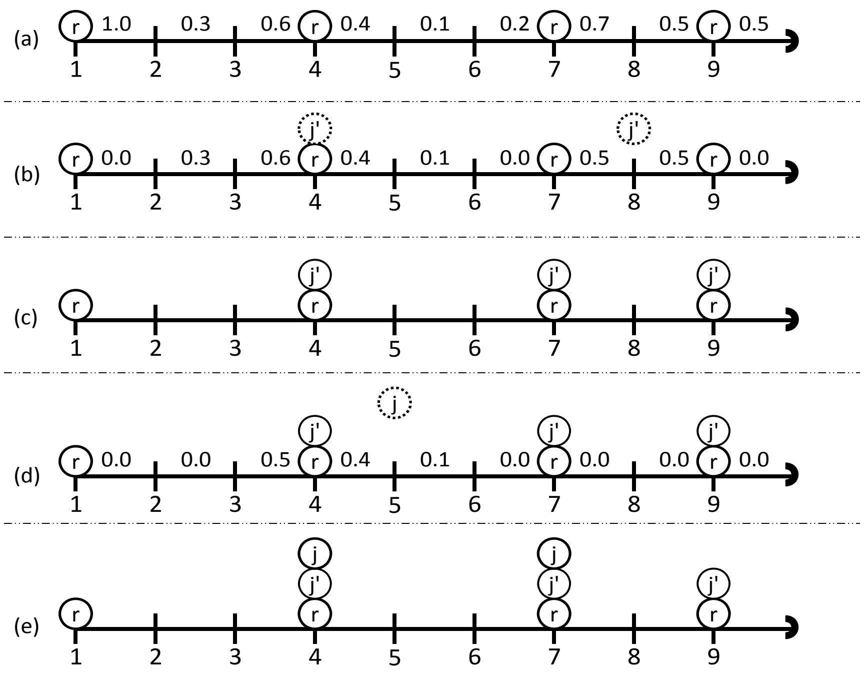

Figure 4-2: This figure demonstrates our LP rounding procedure for the first three successive nodes of the tree, which we call node r (root), node j', and node j. There are 9 time periods in this example. In (a), each circle denotes an order placed for the root node. The values on the interval correspond to the values of y_t^r. In (b), each dotted circle denotes a tentative order placed for node j'. The values on the interval correspond to the values of $y_t^{j'}$. In (c), we do a two sided push for the tentative orders of j', which results in actual orders of j' at periods 4, 7, and 9. In (d), the dotted circle denotes a tentative order placed for node j. The values on the interval correspond to the values of y_t^j. In (e), we do a two sided push for the tentative orders of j, which results in actual orders of j at periods 4 and 7.

Proof. The proof is by induction on $path(j)$, starting from the root. The induction argument uses a monotonicity property of the optimal solution (x, y) mentioned in Lemma 24. Namely, for any time period r and for each pair of nodes j and j' such that $j' \in path(j)$, then $y_r^j \leq y_r^{j'}$.

Base case (root node): For any time interval $[s,t]$ where $\sum_{r=s}^{t} y_r^0 \geq 1$, the interval $(\sum_{r=1}^{s-1} y_t^0, \sum_{r=1}^{t} y_t^0]$ must contain an integer. By construction of the algorithm, an order for the root node must then exist somewhere in $[s,t]$.

Inductive case: Consider node j and any time interval $[s,t]$ where $\sum_{r=s}^{t} y_r^j \geq 1$. Let j' be the parent of j. By the monotonicity property of y, we know $y_t^{j'} \leq y_t^j$ and hence $\sum_{r=s}^{t} y_r^{j'} \geq 1$. Using the induction hypothesis, we know there is an order for node j' in $[s,t]$, namely at period u'. Also, by a similar reasoning as in the base case, we know there is a tentative order for node j in $[s,t]$, namely at period u. We assume u' was chosen to be the closest order of j' to u that is in $[s,t]$. Now since the algorithm opens order of j by a two sided push to the two closest orders of j' in opposite directions, then j will be ordered at u' as well. Thus we have proven the inductive case and the claim follows. □

The correctness of the algorithm follows from Lemma 25 above. Specifically, for each demand point (i,t), it follows from constraints (4.8) and (4.9) in (P-T) that $\sum_{s=1}^{t} y_s^i \geq 1$. Thus, we know that for each node $j \in path(i)$ there is an order placed in $[1,t]$ from Lemma 25. Hence, for each demand point (i,t) there is an order of item i no later then time t that can serve the demand. This implies that the solution is indeed feasible.

Next we analyze the cost of the solution produced by the algorithm. We start by considering the ordering cost. Since the number of orders made is at most $\lfloor \sum_{r=1}^{T} y_r^0 \rfloor$ for the root node and $2\lfloor \sum_{r=1}^{T} y_r^j \rfloor$ for all other nodes j (we make up to two orders for every tentative order of j), the next lemma follows directly.

Lemma 26. *The total ordering cost for the solution by the algorithm is at most* $2\sum_{s=1}^{T} \sum_{j \in \mathcal{T}} K_j y_s^j$.

Finally we analyze the holding cost incurred by the constructed solution by the algorithm. We will show that the total holding cost incurred is at most $\sum_{i=1}^{N} \sum_{t=1}^{T} b_t^i$, the optimal value of (D-T).

Lemma 27. *The total holding cost for the solution by the algorithm is at most* $\sum_{i=1}^{N} \sum_{t=1}^{T} b_t^i$.

Proof. For any demand point (i, t), consider the set of orders s that serve (i, t) fractionally in the optimal solution for (P-T), i.e., $x_{st}^i > 0$. Let s_1 be the earliest of such orders and we define $[s_1, t]$ as the *active interval* for demand (i, t), specifically $x_{s_1 t}^i > 0$ and $\sum_{s=s_1}^{t} x_{st}^i = 1$. Since $x_{s_1 t}^i > 0$, by the complementary slackness conditions, the corresponding dual constraint must be tight, i.e., $b_t^i = H_{s_1 t}^i + \sum_{j \in path(i)} l_{s_1 t}^{ij}$. This expression, combined with the nonnegativity constraints on $l_{s_1 t}^{ij}$, implies that $b_t^i \geq H_{s_1 t}^i$. However, we also assume that the holding cost H_{st}^i is non-increasing in s. It follows that for any time s in the active interval, we have that $b_t^i \geq H_{st}^i$. Hence, it suffices to show that there exists an order for i in the active interval for (i, t). However, by the definition of active interval and constraints (4.8) and (4.9) in (P-T), we have that $\sum_{s=s_1}^{t} y_s^i \geq 1$. Thus, using Lemma 25 for the interval $[s_1, t]$ shows that there exists an order of item i in the active interval of (i, t), as desired. $\square$

By Lemmas 26 and 27, and the fact that the optimal value of (P-T) and (D-T) are both lower bounds to the value of the optimal solution to the tree JRP, we obtain the following result.

Theorem 15. *The LP Rounding algorithm is a 3-approximation algorithm for the tree JRP.*

4.4 Laminar Joint Replenishment Problem

In this section, we study the laminar joint replenishment problem. This is a special case of the submodular JRP, where the setup cost function corresponds to a laminar family. A laminar family $\mathcal{F}$ is a collection of subsets of $\{1, \ldots, N\}$ such that for any $S_1, S_2 \in \mathcal{F}$, either $S_1 \cap S_2 = \emptyset$, $S_1 \subseteq S_2$, or $S_2 \subseteq S_1$. The setup cost of ordering any subset $S \in \mathcal{F}$ is K_S. In the laminar JRP, the cost of ordering a set of items S in any

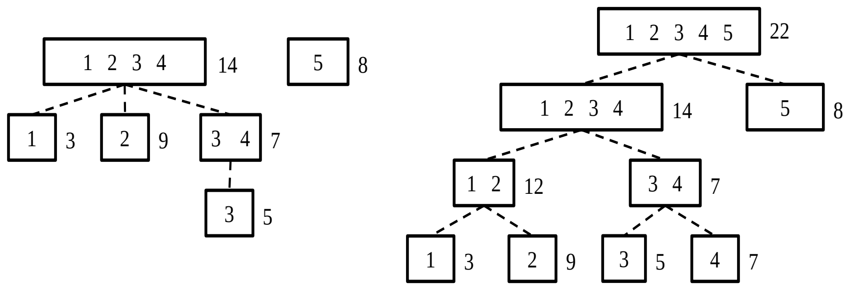

Figure 4-3: These two graphs represent two laminar families that are equivalent. Each box represents a machine that can produce the item types inside of it. The number to the right of each box is the fixed cost of the machine. The dashed lines denote which machines are strictly more capable than the others. The left representation is a typical input to this problem, and the right representation is a transformation to a binary tree graph. In both representations, the cost of ordering any subset of items is the cost of the cheapest machine containing those items.

time period is the cost of the smallest element of $\mathcal{F}$ that contains S. More formally, $K(S) = \min_{F \in \mathcal{F}} K_F$ s.t. $S \subseteq F$.

A laminar family can be represented by a tree graph such as in Figure 4-3. The root node consists of all item types. Then the item types in each node are split amongst the children nodes. In order for the laminar family to correspond to a proper submodular joint setup cost function, then clearly (i) the cost of every node is at least the cost of each of its children (monotonicity) and (ii) the cost of every node is no more than the total cost of its children (submodularity). Without loss of generality, one can always represent a laminar family with a binary tree as in Figure 4-3. We will make this assumption for the remainder of this section.

4.4.1 Dynamic Programming Formulation

We now give a polynomial-size dynamic programming formulation to solve the laminar JRP. For each node/subset $F \in \mathcal{F}$ such that $|F| \geq 2$, let $c_1(F)$ and $c_2(F)$ be its children nodes. Let $J(F, s, t)$ be the optimal cost of serving the demands from s to t for all of the item types in F. When computing $J(F, s, t)$, we assume that the item types in F have been ordered at time s. Thus, the optimal overall cost is $J(\{1, \ldots, N\}, 0, T)$, and we are assuming without loss of generality that there is an order at artificial time 0. This will require that we define $d_0^i = 0$ and $H_{0t}^i = \infty$ for all i and t. Now we present the dynamic programming equation when $|F| \geq 2$,

$$J(F,s,t) = \min(J(c_1(F),s,t) + J(c_2(F),s,t), \min_{u=s+1,\dots,t} J(F,s,u-1) + K_F + J(F,u,t)).$$

The first term in the outer min is the case where no extra order of F occurs between s and t. The other case is that we order F at some point u between s and t. The cost in this case is the cost of F between s and $u-1$ plus the fixed order cost at u plus the remaining cost of F from u to t. When $|F| = 1$, we account for the holding cost of the corresponding item. To solve this base case, we essentially solve a standard economic lot sizing problem as in Wagner and Whitin [89]. The dynamic programming equation is given by

$$J(F,s,t) = \min(\sum_{v=s}^{t} H_{sv}^F, \min_{u=s+1,\dots,t} J(F,s,u-1) + K_F + J(F,u,t)).$$

The first term in the outer min denotes the case where no further orders of F are placed between s and t, and thus we now know the holding costs for all the demand points between s and t for the item type corresponding to F. The second case is when an extra order of F is placed in between s and t and the problem is then decomposed into two subintervals, with an additional fixed ordering cost.

In order to solve the complete dynamic program, we start at the leaves of the laminar tree and work up towards the root. At each leaf, solve the intervals of length 1, 2, and so on. One can easily show that the dynamic program provides an optimal solution due to the natural decomposition of the tree: the cost of $J(F,s,t)$ only depends on the cost of the children of F from s to t. In addition, we account for all the orders as we place them, and account for the holding costs exactly when we stop placing orders in a given interval. Finally, since $\mathcal{F}$ can have at most size $2N$, then there are at most $O(NT^2)$ values of J. Since each J is computed in $O(T)$ time, this gives an overall runtime of $O(NT^3)$. Thus, we obtained the following theorem.

Theorem 16. *There exists a polynomial-time dynamic programming algorithm to solve the laminar JRP.*

4.5 Cardinality Joint Replenishment Problem

In this section, we consider the cardinality JRP, which is a special case of the submodular JRP where the joint setup cost is a function of the cardinality of the subset of item types being ordered. By the submodularity property, this function must be concave. Formally, we let $g(k)$ be a nondecreasing, concave function which denotes the cost of ordering k item types in any given period. Thus the cost of ordering the item types in S in any given period is $K(S) = g(|S|)$. In the following subsection, we will give a polynomial-size formulation for the cardinality JRP.

4.5.1 A Linear Program

In the following formulation, we let x^i_{st} have the same interpretation as formulation (IP) for the submodular JRP. In addition, the variable z^i_s will be 1 if an order with item type i is placed in period s and 0 otherwise. Finally, the variable q^k_s will be 1 if there was an order in period s of size at least k and 0 otherwise.

$$\text{minimize} \sum_{s=1}^{T}\sum_{k=1}^{N}(g(k)-g(k-1))q^k_s + \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{s=1}^{t} H^i_{st}x^i_{st} \tag{IP-C}$$

$$\text{subject to} \sum_{s=1}^{t} x^i_{st} = 1 \qquad i=1,\dots,N, t=1,\dots,T \tag{4.12}$$

$$x^i_{s,t} \le z^i_s \qquad i=1,\dots,N, s=1,\dots,T, t=s,\dots,T \tag{4.13}$$

$$q^{k+1}_s \le q^k_s \qquad k=1,\dots,N-1, s=1,\dots,T \tag{4.14}$$

$$\sum_{k=1}^{N} z^k_s \le \sum_{k=1}^{N} q^k_s, \quad s=1,\dots,T \tag{4.15}$$

$$x^i_{st}, z^i_s, q^k_s \in \{0,1\}, i=1,\dots,N, s=1,\dots,T, t=s,\dots,T, k=1,\dots,N$$

Lemma 28. *(IP-C) is a correct integer programming formulation for the cardinality JRP.*

Proof. First, we show that there is a one-to-one correspondence between solutions of (IP-C) and the cardinality JRP. Given a solution to (IP-C), we simply order and serve demand according to the variables x^i_{st}. Conversely, given a solution to the cardinality JRP, we let x^i_{st} be defined as it is served in the solution. For every $s = 1, \dots, T$, if there are k item types ordered in s, we let $z^i_s = 1$ for those k item types and otherwise set $z^i_s = 0$. Also we set $q^1_s = \dots = q^k_s = 1$ and $q^{k+1}_s = \dots = q^N_s = 0$. Next we check that all constraints are satisfied. First, since every demand point is served by some order, constraint (4.12) is clearly satisfied. Next, a demand point of item type i cannot be served without an order of item type i being placed, then constraint (4.13) is satisfied. By construction of q^k_s above, constraint (4.14) is also satisfied. By construction, constraint (4.15) is satisfied with equality.

Now we just need to show that the cost of a solution to (IP-C) correctly models the cost of a solution of the cardinality JRP. The holding cost is modeled by the second term of the objective of (IP-C), and is clearly accurate for some solution x. For a solution x, we know that the (IP-C) will set as few of the z variables as possible to 1 since this will require as few as possible q variables to be 1. This property holds due to the fact that $g(k) - g(k-1)$ is always nonnegative due to the monotonicity of g. Constraint (4.14) ensures that if k item types are ordered in period s, then only $q^1_s = \dots = q^k_s = 1$. Thus the fixed order cost in this time period is exactly equal to $g(k)$ due to the telescoping sum. Thus, the overall cost of this formulation correctly captures the actual cost. $\square$

Let (P-C) be the natural LP-relaxation of (IP-C) where the binary constraints on the variables are relaxed to nonnegativity constraints. Next, we will show that the formulation (P-C) is equivalent to the original formulation (P) in the case of the cardinality JRP. We denote Z_{P-C} and Z_P as the optimal values of (P-C) and (P), respectively.

Lemma 29. *For the cardinality JRP, (P-C) is equivalent to (P), i.e., $Z_{P-C} = Z_P$.*

Proof. The proof follows the same outline as in Lemma 24. First we show how to convert an optimal solution $(\hat{x}, \hat{z}, \hat{q})$ of (P-C) to a solution (x, y) of (P). We first set $x = \hat{x}$. Next we set the values of y. Fix a time period s, and we first establish an ordering $\sigma(i)$ of the items from the optimal solution of (P-C) so that $\hat{z}_s^{\sigma(1)} \geq \hat{z}_s^{\sigma(2)} \geq \ldots \geq \hat{z}_s^{\sigma(N)}$. Next define the sequence of sets $S(i) := \{\sigma(1), \ldots, \sigma(i)\}, i = 1, \ldots n$. We let $y_s^{S(N)} = \hat{z}_s^{\sigma(N)}$, $y_s^{S(i)} = \hat{z}_s^{\sigma(i)} - \hat{z}_s^{\sigma(i+1)}$ for $i = 1, \ldots, n-1$, and $y_s^S = 0$ for all other set S. It is easy to check that this solution is feasible for (P), and the objective value is also the same by construction. Thus we have shown that $Z_{P-C} \geq Z_P$.

For the converse, we show that we can convert an optimal solution $(\hat{x}, \hat{y})$ to (P) to a solution (x, z, q) to (P-C) with the same cost. Let $(\hat{x}, \hat{y})$ satisfy the conditions of Lemma 23 , which implies that for each t, there is an ordering of N such that the only sets that can be positive are $\hat{y}_t^{\emptyset}, \hat{y}_t^{S_1}, \ldots, \hat{y}_t^{S_N}$ where $\emptyset \subset S_1 \subset S_2 \subset \ldots \subset S_N = \mathcal{N}$. We first set $x = \hat{x}$. We now let $z_s^i = \max\{x_{st}^i : t = s \ldots T\}$ and $q_t^i = \sum_{k=i}^n \hat{y}_t^{S_k}$. One can check that all of the constraints of (P-C) are satisfied. The holding costs of the solutions are clearly the same. The ordering cost of the the solution (x, z, q) is:

$$\begin{aligned}
\sum_{t=1}^{T}\sum_{i=1}^{N}(g(i) - g(i-1))q_t^k &= \sum_{t=1}^{T}\sum_{i=1}^{N}(g(i) - g(i-1))\sum_{k=i}^{N}\hat{y}_t^{S_k} \\
&= \sum_{t=1}^{T}\sum_{k=1}^{N}\sum_{i=1}^{k}(g(i) - g(i-1))\hat{y}_t^{S_k} \\
&= \sum_{t=1}^{T}\sum_{k=1}^{N}\hat{y}_t^{S_k}\sum_{i=1}^{k}(g(i) - g(i-1)) \\
&= \sum_{t=1}^{T}\sum_{k=1}^{N}\hat{y}_t^{S_k}g(k)
\end{aligned}$$

Thus the two solutions have equal cost. Since $(\hat{x}, \hat{y})$ was optimal for (P), then we have shown that $Z_{P-C} \leq Z_P$ which completes the proof. $\square$

Now we consider the dual of (P-C), which we call (D-C). Let b_t^i, l_{st}^i, v_s^k and w_s^i be the dual variables corresponding to constraints (4.12), (4.13), (4.14), and (4.15) in

(P-C), respectively. The dual of (P-C) is:

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{N}\sum_{t=1}^{T} b_t^i && && \text{(D-C)}\\
\text{s.t.} \quad & b_t^i \le H_{st}^i + l_{st}^i, && i=1,\dots,N, t=1,\dots,T, s=1,\dots,t && (4.16)\\
& v_s^k - v_s^{k-1} + \sum_{i=k}^{N} w_s^i \le g(k)-g(k-1), k=2,\dots,N-1, s=1,\dots,T && && (4.17)\\
& -v_s^{N-1} + w_s^N \le g(N)-g(N-1), && s=1,\dots,T && (4.18)\\
& v_s^1 + \sum_{i=1}^{N} w_s^i \le g(1), && s=1,\dots,T && (4.19)\\
& \sum_{t=s}^{T} l_{st}^i - \sum_{k=i}^{n} w_s^k \ge 0, && i=1,\dots,N, s=1,\dots,T && (4.20)\\
& l_{st}^i, v_s^k, w_s^i \ge 0, i=1,\dots,N, t=1,\dots,T, s=1,\dots,t, k=1,\dots,N-1
\end{aligned}
$$

Following the same strategy as was used in the previous section, we will use (D-C) to bound the holding cost incurred by our algorithm.

4.5.2 Labeling Algorithm

We now describe the LP rounding algorithm for the cardinality JRP. This algorithm will rely on the optimal solution to (P-C), which we denote by (x, z, q). For each item type i, we will partition the interval $(1, T+1]$ into subintervals, each with a weight of exactly 0.5. For every period $t = 1, \dots, T$, the weight of the region $(t, t+1]$ will be exactly z_t^i, which is spread uniformly across this time period. The partition of $(1, T+1]$ is done by myopically adding subintervals of weight 0.5 starting from time 1. Specifically, let P denote the collection of subintervals. Initially set $a = 1$. Let b be the time in $(1, T+1]$ where the weight of $(a, b]$ is exactly 0.5. Add $(a, b]$ to P, set $a := b$ and repeat until no further subintervals can be added. See Figure 4-4 for an example of this procedure.

Now let P denote the set of all (sub)intervals generated using the procedure pre-

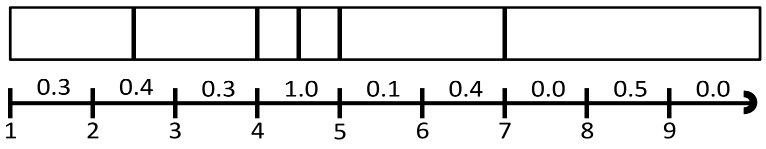

Figure 4-4: This figure represents one item type i for a problem with 9 time periods. The values on the time interval correspond to the values of z_s^i. The partitioned rectangle above the time interval denotes the corresponding subintervals, each with a weight of 0.5, that would be added to P; namely, $(1, 2.5]$, $(2.5, 4]$, $(4, 4.5]$, $(4.5, 5]$, $(5, 7]$, and $(7, 9]$.

viously described on all the item types. We denote interval j of P by $(a_j, b_j]$. For convenience, we assume that P is sorted by the values of b_j, with ties broken arbitrarily. The item corresponding to interval j is denoted by i_j. Finally, each interval j will be given a label l_j, which can take the values from $1, \ldots, N$. The labels will correspond to how large an order was after the item type i_j corresponding to interval j was added to an order. Specifically, we will start by labeling intervals with the label 1, then label 2, and so on. For every round of labeling k, we process P in increasing order of the b_j values. If the current interval j in P does not have a label, and there does not exists an order of size k in $(a_j, b_j]$, then i_j is ordered at the latest order of size $k-1$ in $(a_j, b_j]$ and l_j is set to k. Note that in labeling round 1, when an order is placed it always occurs at b_j since "orders of size 0" exist everywhere. Finally, after all the labeling, we round every order at at time τ to $\lfloor \tau - \epsilon \rfloor$. All the demands are then served myopically. See Figure 4-5 for an example of the algorithm. Below is the pseudocode for the algorithm.

4.5.3 Analysis

We will use the labels of the orders made by the algorithm to bound its cost versus the optimal value of (P-C). We begin by proving several properties of the labeling algorithm, including its correctness.

Lemma 30. *Every interval $j \in P$ receives a label.*

Proof. Assume there exists $j \in P$ without a label. Let k be the size of the largest

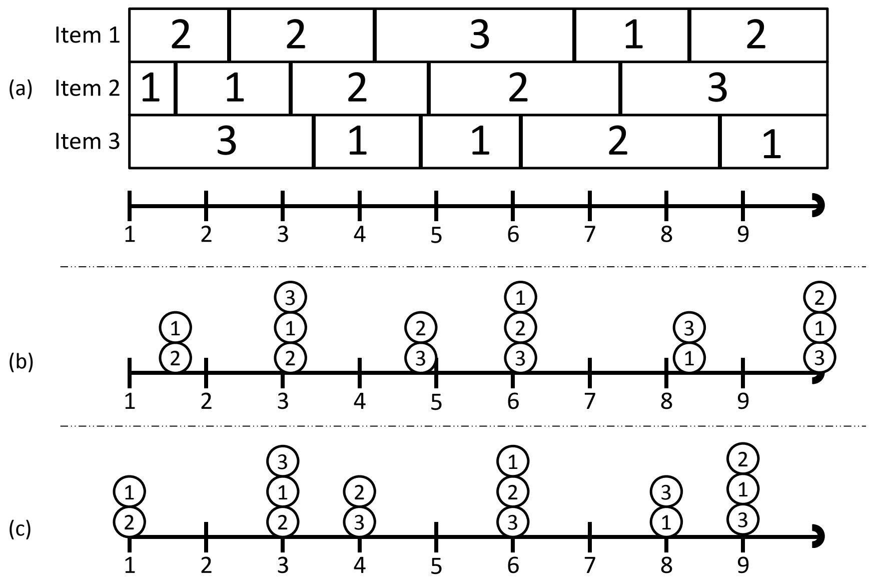

Figure 4-5: This figure shows an example of the LP rounding algorithm. In (a), each rectangle correspond to an interval in P. Each row of intervals corresponds to an item type. The intervals were generated using the partition procedure depicted in Figure 4-4. The number in each rectangle/interval corresponds to the label it was given using Algorithm 2. In (b), we show how these labels correspond to orders. The number inside each circle denotes the item type that was ordered. The height of each circle corresponds to its label. In (c), we show how to round each order down to the nearest integral point.

Algorithm 2: Labeling Algorithm for cardinality JRP

```
Solve (P-C)
// Generate intervals of $P$

for $i \in \mathcal{N}$ do
    Initialize $a$:=1 and let the weight of period $t$ be $z^i_t$ on the region $[t, t+1]$
    Choose $b$, if it exists, so that weight of $(a, b]$ is 0.5
    Add $(a, b]$ to $P$ and repeat with $a := b$
// Generate orders and labels

for $k \leftarrow 1$ to $N$ do
    for $j \leftarrow 1$ to $|P|$ do
        if $j$ is unlabeled and $\nexists$ an order of size $k$ in $(a_j, b_j]$ then
            Order $i_j$ at the latest order of size $k-1$ in $(a_j, b_j]$
            Set label $l_j = k$
Round ever order at time $\tau$ to $\lfloor \tau - \epsilon \rfloor$
Serve every demand point $(i, t)$ from the latest order up to time $t$ that includes
item type $i$;
```

order in $(a_j, b_j]$. We know $k \neq N$ or else i_j is in the order and j was labeled. We also know that this order existed in the $(k+1)^{st}$ labeling round of the algorithm by construction. This implies that j should have been added to that order, or another order of size k in $(a_j, b_j]$, according to the algorithm. Either way, j would be labeled, which is a contradiction. □

As in the analysis of the algorithm for tree JRP, we define the *active interval* for demand point (i, t) to be $[s_1, t]$, where s_1 is be the earliest time period where $x^i_{st} > 0$.

Lemma 31. *Every demand (i, t) is served from its active interval $[s_1, t]$.*

Proof. Consider a demand point (i, t) and its active interval $[s_1, t]$. From feasibility of x, we know that there must be at least one interval j of item type i in P such that $(a_j, b_j] \subseteq [s_1, t+1]$. Therefore, by Lemma 30, we must have had an order of item type i in $[s_1, t+1]$. Since the orders are rounded down in time, we guarantee that the order is in $[s_1, t]$, which completes the proof. □

Lemma 32. *For any time $\tau \in (1, T+1]$ and $k \in 1, \ldots, N$, there are at most 2 intervals in P that contain τ and are both labeled k.*

Proof. First we show that we cannot have two intervals j and j' with labels k such that $(a_j, b_j] \subseteq (a_{j'}, b_{j'}]$. If $b_j < b_{j'}$, then $(a_j, b_j]$ is processed first. Then when we are processing j' in the k^{th} labeling round, there exists an order of size k in the interval for j'. Therefore, the algorithm would skip interval j' in the k^{th} round. If $b_j = b_{j'}$, then there are two cases depending on what is processed first. If j is processed first, then the same logic holds. If j is processed second, then it must be the case that the order where $i_{j'}$ was added occurred in $(a_{j'}, a_j]$, or else there is an order size k in j's interval and it would be skipped. However, since i_j was added to an order of size $k-1$ in its interval, then it must be that j' could have been ordered at a later point in time which is what the algorithm mandates.

Now assume for contradiction that there are three orders j, j' and j'' with label k, all of which contain some τ in their interval. From the previous argument, we can assume that $a_j < a_{j'} < a_{j''} \leq \tau \leq b_j < b_{j'} < b_{j''}$. Observe that the order where i_j was added must occur in $(a_j, a_{j'}]$, or else there would be a size k order in the interval of j', and thus it would get passed over in the k^{th} labeling round. Now consider the following two cases of where the order of j' occurred. If it occurs in $(a_{j'}, b_j)$, this implies that j did not order at the latest possible order of size $k-1$. If the order of j' occurred within $[b_j, b_{j'}]$, this implies that j'' is processed when there is an order of size k in its interval, and so the algorithm would skip it in the k^{th} labeling round. Therefore, we cannot give a label of k to j', which is a contradiction. $\square$

Using the previous lemmas, we are now we are ready to prove the main result.

Theorem 17. *The labeling algorithm is a 5-approximation algorithm for the cardinality JRP.*

Proof. We first bound the holding cost for each demand (i, t) by b^i_t, the dual variable of (D-C). By Lemma 31, (i, t) is served by an order s such that $x^i_{st} > 0$. From complementary slackness, this implies that $b^i_t = H^i_{st} + l^i_{st}$. Since $l^i_{st} \geq 0$, then the holding cost paid by (i, t) is $H^i_{st} \leq b^i_t$. This implies the holding cost is at most the optimal cost, since the sum of b^i_t over all demand points gives the dual objective.

Now we proceed to bound the ordering cost of our solution. For every interval $j \in P$, we assigned it a label $l_j = k$ for some k. This means that item type i_j is the k^{th} item type added to some (unrounded) order in the interval $(a_j, b_j]$. Thus, this interval needs to pay $g(k) - g(k-1)$ in order to account for itself. We therefore want to take at most the k^{th} slice, $g(k) - g(k-1)$, of the ordering costs corresponding to interval j. This amounts to using the variable q_t^k wherever $z_t^{i_j} > 0$. We can also use lower slices, i.e., $q_t^1, \ldots q_t^{k-1}$ due to the concavity of g. If we are able to do this such that no slice gets counted for twice, then we ensure that we can pay for at least half the ordering costs, since the weight of every interval is 0.5, and by Lemma 30 every interval corresponds to an item type being ordered. The exact charging scheme is as follows. For every $\tau \in (1, T+1]$, we give the cost of $q_\tau^1 g(1)$ to the two item types whose interval contains τ and has the smallest labels. Then we give the cost of $q_\tau^2(g(2) - g(1))$ to the two item types whose intervals contain τ and have the third and fourth smallest labels and so on. From Lemma 32, this procedure guarantees that every interval j is given slices of g across $(a_j, b_j]$ that are at most the l_j^{th} slice. Thus, we need twice the optimal fractional ordering cost to pay for half of the marginal ordering cost incurred by any interval $j \in P$ in our solution, and so we need at most 4 times the optimal fractional ordering cost in total to pay for the orders.

Combining the two bounds yields the result. □

4.6 Conclusion

We believe that this work advances the existing research for an important class of interesting inventory management problems with non-stationary demand. The submodular JRP and the special cases we consider can capture a wide variety of real world problems and allows for substantially more modeling flexibility. Since most variants of the JRP are NP-hard, it is intractable to compute the optimal solutions efficiently. Our algorithms are computationally efficient and provide theoretical worst case guarantees. Moreover, we provided strong and polynomial size LP formulations for the tree and cardinality JRP which may be useful for solving the IP formulations in practice. One major open question which we have left unanswered is whether or not there exists a constant factor approximation algorithm for the submodular JRP.

Appendix A

Chapter 2 Appendix

A.1 Examples

Example 3. *Consider the online algorithm where we solve the offline optimal solution with respect to the first k customers, and the decisions for the first $k-1$ customers are fixed, i.e., they cannot be changed from what we have already decided. Then this will result in most customers being rejected if each customer on his own is not enough to warrant production.*

For example, consider the To Build or Not To Build problem as explained in Example 1. If the rejection cost of each customer is less than K, then clearly the algorithm we just described will reject everyone in an online fashion. The competitive ratio will be $R(U)/K$ which can be arbitrarily poor.

Example 4. *Consider the ELS problem with online customer selection from Section 2.3.2. Let $h = 1, K = 11$, and $r = 10$. Consider the input sequence $I_1 = (1, 8), I_2 = (1, 14), I_3 = (1, 1)$, and $I_4 = (100, 1)$, where each tuple represents demand quantity and due date, respectively. Let the set of potential order dates be $\mathcal{Q} = \{1, 2, \ldots, 15\}$. One can easily show that $\mathcal{A}^* = \{1, 3, 4\}$, $P(\mathcal{A}^*) = 19$, $\mathcal{R}^* = \{2\}$, and $R(\mathcal{R}^*) = 10$. The Copycat Algorithm will lead to $\mathcal{R}^C = \{1, 3\}$ with $R(\mathcal{R}^C) = 20$. StablePair will only have $\mathcal{R}^S = \{1\}$ with $R(\mathcal{R}^S) = 10$. Note that $\mathcal{R}^C \cap \mathcal{R}^* = \mathcal{R}^S \cap \mathcal{R}^* = \emptyset$. Thus,*

$$R(\mathcal{R}^S) < P(\mathcal{A}^*) + R(\mathcal{R}^C \cap \mathcal{R}^*) < R(\mathcal{R}^C) < P(\mathcal{A}^*) + R(\mathcal{R}^*)$$

This demonstrates that there are cases in which StablePair can be strictly less conservative than Copycat in that it accepts customers that were rejected by Copycat.

Example 5. *Consider the ELS problem with online customer selection from Section 2.3.2. Let* $h = 1, K = 11$*, and* $r = 10$*. Let the set of potential order dates be* $\mathcal{Q} = \{1, 2, \ldots, 15\}$*. Consider the input sequence* $I_1 = (2, 4), I_2 = (1, 11)$*, and* $I_3 = (100, 1)$*, where each tuple represents demand quantity and due date, respectively. One can easily see that* $\mathcal{A}_1^* = \{1\}$*,* $\mathcal{A}_2^* = \{1, 2\}$*, and* $\mathcal{A}_3^* = \{1, 3\}$*. This implies that* $\mathcal{A}_3^C = \{1, 2, 3\}$*. Here we see that Copycat "regrets" accepting customer 2 after stage 3 since the optimal offline solution in stage 3 rejects customer 2.*

Example 6. *Consider the ELS problem with online customer selection from Section 2.3.2. Let* $h = 1, K = 11$*, and* $r = 10$*. Let the set of potential order dates be* $\mathcal{Q} = \{1, 2, \ldots, 10\}$*. Consider the input sequence* $I_1 = (2, 10)$ *and* $I_2 = (1, 3)$*, where each tuple represents demand quantity and due date, respectively. Clearly* $\mathcal{A}_1^* = \mathcal{A}_2^* = \{1\}$ *and the optimal offline cost is* $C^*(\{1, 2\}) = 11 + 10 = 21$*. However,* $\mathcal{A}_2^S = \{1, 2\}$*. StablePair accepts customer 1 with stable pair* $(\mathcal{Q}, \{1\})$*, where* $\mathcal{Q}$ *represents all possible time periods to create a setup. However, unlike the optimal offline solution, StablePair also accepts customer 2 with stable pair* $(\{3\}, \{1, 2\})$*. If we can only create an order at time 3, then the optimal strategy is to accept both customers 1 and 2. If we can order at any time, then we would only order at time* 10 *and reject customer 2.*

Example 7 (Polymatroid Optimization)**.** *Let* U *be a ground set and* $f(\cdot)$ *be a* rank function*, i.e., a nondecreasing submodular function with* $f(\emptyset) = 0$*. Then the polyhedron* $F(U, f) = \{\mathbf{x} \in \mathbb{R}^{|U|} : \sum_{i \in T} x_i \leq f(T),\ T \subseteq U,\ \mathbf{x} \geq 0\}$ *is referred to as a* polymatroid*. Now for each* $i \in U$*, let* $g_i : \mathbb{R} \to \mathbb{R}$ *be a concave function. Then for any set* $T \subseteq U$ *the production cost function is defined as*

$$P(T) = \max \sum_{i \in T} g_i(x_i) \text{ s.t. } \mathbf{x} \in F(T, f).$$

[43] showed that $P(T)$ *is indeed nondecreasing and submodular.*

An application of this result, as shown in [43], is the stationary joint replenishment problem. *In this problem the "customers" are actually products. Each product k arrives with $I_k = \{d_k, h_k\}$ where d_k is the stationary continuous demand rate and h_k is the holding cost rate. The rejection cost r_k is the revenue rate generated by producing product k. Demand for accepted products is satisfied by a sequence of orders, where an order for the item types in $T \subseteq U$ has fixed costs $K(T)$. $K(\cdot)$ is assumed to be a rank function. All demands are served by their nearest setup and also incur the appropriate holding costs. In [31] it was shown that the optimal policy for the stationary JRP can be approximated well by only considering "power-of-two" (Po2) polices. Indeed, the problem of finding the best Po2 policy then reduces to a polymatroid maximization problem with a separable concave objective. The best of such Po2 policies is at most 2% suboptimal. [43] also shows a similar result for the continuous one-warehouse multi-retailer problem. In our online customer selection model, products arrive online to a supplier, who decides whether to accept or reject them, and then computes the best Po2 policy to serve them.*

Example 8. *Let $h = 1, r = 2M$, and $K = 2M^2 + 1$. Let the set of potential order dates be $\mathcal{Q} = \{1, 2\}$. Let $S = \{1, 2, 3\}$ and $I_1 = (M, 2), I_2 = (1, 1)$, and $I_3 = (100K, 2)$, where the tuples represent quantity and due date, respectively. Observe that $\mathrm{OPT}(U_1)$ will reject customer 1 since the setup cost is greater than the rejection cost of $2M^2$. Then $\mathrm{OPT}(U_2)$ will accept both customers 1 and 2, and the total cost will be $2M^2 + 1 + M$, which is less than rejecting both customers at cost $2M^2 + 2M$. Finally, $\mathrm{OPT}(U_3)$ will clearly accept customers 1 and 3, while rejecting customer 2. The optimal offline cost will be $C^*(U) = P(\{1 \cup 3\}) + R(\{2\}) = 2M^2 + 1 + 2M$. The cost of Copycat and StablePair will both be $C(U) = P(\{2 \cup 3\}) + R(\{1\}) = 2(2M^2 + 1) + 2M^2 = 6M^2 + 2$. Thus, the competitive ratio is asymptotically close to 3 as M goes to ∞.*

Example 9. *Let $h = 0$ and $d_k = 1$ for all customers. Let $I_1 = (0, 1, T)$. Clearly a good online algorithm must reject customer 1 or else the competitive ratio would be K/r after the first arrival. Now we repeat the following algorithm to generate a worst case adversary.*

0. Initialize $s := 0$, $t := T/2$, $u = T$, and $k = 2$. Let $I_1 = (0, 1, T)$.

1. Let $I_k = (s, 1, t - \epsilon)$.

2. If online algorithm accepts k, then go to Step 3. Else, go to Step 4.

3. Set $s := t$, $t := s + (u - s)/2$, and $u := u$. Go to Step 1.

4. Set $s := s$, $t := s + (u - s)/4$, and $u := s + (u - s)/2$. Go to Step 1.

Let this adversary generate N arrivals and let n be the number of customers that the algorithm accepts. From the construction, it is clear that the algorithm must satisfy each customer with separate orders. Thus, the cost incurred by the algorithm is $nK + (N - n)r$. A good feasible solution to the sequence would be the exact opposite of what the algorithm did. By construction, all the customers that the algorithm rejected (at least one since it must reject customer 1) can be satisfied by one order. Thus the cost of this feasible solution is $K + nr$. The competitive ratio is then at least

$$\frac{nK + (N - n)r}{K + nr} \geq \min\{\frac{Nr}{K}, \frac{K - r}{r}\}.$$

For a given K and r, if we take N to be large enough, then the competitive ratio is at least $K/r - 1$ which is not a constant. One can extend the proof to discrete time by making T large enough.

Example 10. *Consider a single facility with cost f. Let $r = \sqrt{2}f - \epsilon$. Let the first customer that arrives request 1 unit at the facility, so the service cost would be 0. Then StablePair with scaling factor $\alpha = \frac{1}{\sqrt{2}}$ rejects customer 1. Let the second customer be identical to the first, and so StablePair with scaling accepts customer 2. Clearly the optimal solution is to serve both customers and incur cost f, while StablePair incurs a cost of $(1 + \sqrt{2})f - \epsilon$. Therefore, the competitive ratio can be arbitrarily close to $1 + \sqrt{2}$ as ϵ goes to 0.*

A.2 Proofs

Proof of Lemma 9. Assume there is a setup interval $[a, b]$ in the solution to $P(Q, T)$ such that $b - a > r/h$. By the ZIO property, there must be an order at a. This means there exists a customer k with due date at time b that pays a per unit holding cost greater than r. This implies that the optimal solution of $\text{OPT}(Q, T)$ is better off rejecting k and keeping everything else the same, which is a contradiction to the stability of (Q, T). $\square$

Proof of Lemma 10. Assume that $[a, b]$ does not contain the due date of any customers in $\mathcal{A}^*$. By the stability of (Q, T), the rejection cost of the customers served by the order at a are greater than the production costs incurred in the interval $[a, b]$. This implies that $\mathcal{A}^*$ could be augmented to include all customers having due dates in $[a, b]$ without increasing the overall cost. Since we chose $\mathcal{A}^*$ to be maximal, then this is a contradiction. $\square$

Proof of Theorem 6. We will show that $\mathcal{A}^C \subseteq \mathcal{A}^*$ and thus $P(\mathcal{A}^C) \leq P(\mathcal{A}^*)$. Since we will be using the 2-competitive heuristic of [2] for the online ELS problem, then the production costs will be at most $2P(\mathcal{A}^C) \leq 2P(\mathcal{A}^*)$. From Theorem 3, this gives a 3-competitive algorithm.

In fact, we will show that there is a sequence of solutions such that $\mathcal{A}_1^* \subseteq \ldots \subseteq \mathcal{A}_N^*$ which implies $\mathcal{A}^C \subseteq \mathcal{A}^*$. Consider a customer k in $\mathcal{A}_k^*$. Let $k' \geq k$ be the first k' such that $k \notin \mathcal{A}_{k'}^*$. Then the our production horizon can be partitioned into $T_1 = [t_1, t_k]$ and $T_2 = (t_k, t'_k]$. No customers with due dates in T_2 are served by an order in T_1 since customer k was rejected and the rejection cost per unit is the same for all customers. Therefore, the offline optimal solution can be decoupled to solving the respective problems for the customers with due dates in T_1 and T_2 and then merging the solutions together. Since the due dates of the customers are chronological, then the customers corresponding to T_1 are exactly U_k. Therefore, we know there is an optimal offline solution for U_k that accepts k, and thus we can use that to find an optimal solution to $\text{OPT}(U_{k'})$ that accepts k. $\square$

Proof of Lemma 5. Let (Q_k, T_k) be the stable pair that accepted customer k. Consider the order that serves customer k in the ELS production plan for $P(Q_k, T_k)$, and call this order date t. Define $Q = \{t\}$ and T' to be all the customers that are served by the order at time t in $P(Q_k, T_k)$. Clearly $(Q, T') \subseteq (Q_k, T_k)$ and $k \in T'$. The first property holds by construction. Note that (Q, T') is a stable pair since (Q_k, T_k) is stable. The stability property from (2.1) directly implies that $R(T') \geq K + \sum_{j \in T'} h(t_j - t)d_j$. Now let $T = \{j \in U_k | t_j \in [t, t + r/h]\}$, which is exactly the second property. Lemma 9 implies that $T' \subseteq T$. It follows that for all $j \in T$, $rd_j \geq h(t_j - t)d_j$. Therefore, the rejection costs of T are also greater than the production costs $P(Q, T)$, i.e. $R(T) \geq K + \sum_{j \in T} h(t_j - t)d_j$, which is the third property. Properties 1, 2, and 3 are sufficient conditions for (Q, T) to be a stable pair. Thus (Q, T) is a stable pair with $k \in T$ that satisfies all three properties. The other direction is trivial. $\square$

Proof of Theorem 8. The proof is very similar to the proof of Lemma 12, but requires a more involved construction and analysis. Let $s_1, \ldots, s_m$ denote the times of the orders in the optimal production plan for $P(\mathcal{A}^*)$. The cost of the optimal production plan can be decomposed into the total setup costs, K^*, and the total holding costs, H^*. Let $\mathcal{A}^i$ simply denote the customers with item type i that are in $\mathcal{A}$. For each customer $k \in \mathcal{A}^i$, let a_k be the order date that serves customer k in the stable pair solution that StablePair used to accept customer k, and let $\hat{T}^i$ denote the set of these order dates sorted from earliest to latest. Construct $T^i \subseteq \hat{T}^i$ by processing $\hat{T}^i$ in order from earliest to latest, and remove any order date that is within r^i/h^i periods of the previous order date that was not removed. Now consider the following two sets, X^i and Y^i. The set X^i is defined to be the set of all type i customers that have due date within $[t, t + r^i/h^i]$ for some $t \in T^i$. Note that X^i is not necessarily contained in the set $\mathcal{A}^i$. The set Y^i is then defined to be $\mathcal{A}^i \backslash X^i$. For convenience, let us define $\Delta = \max_i r^i/h^i$ and $\delta = \min_i r^i/h^i$.

To construct a solution that serves all the customers in $\mathcal{A}$, we first create the same sequence of orders as in $P(\mathcal{A}^*)$ in periods $s_1, \ldots, s_m$ and incur setups costs of K^*. Note that the item orders are replicated as well. All the customers in $\mathcal{A} \cap \mathcal{A}^*$ are then served in the the same way as in the production plan for $P(\mathcal{A}^*)$, and thereby incur holding costs of at most H^*. Now we create a sequence of duplicate orders shifted back by time Δ, i.e. at periods $s_1 - \Delta, \ldots, s_m - \Delta$, and incur another K^*.

Next, consider each item type i separately. Assume for now that for each order date $t \in T^i$, there exists an order s_j in the production plan for $P(\mathcal{A}^*)$, such that either $s_j \in [t - \Delta, t]$ or $s_j - \Delta \in [t - \Delta, t]$. Moreover, order s_j (or $s_j - \Delta$) includes item i. Assuming this property holds, one can bound the holding costs for the customers in $\mathcal{A}^i \cap \mathcal{R}^*$. Specifically, the property ensures that all type i customers with due dates in $[t, t + r^i/h^i]$ can be served with holding cost at most $(\Delta + r^i/h^i)h^i$ per unit. By definition of X^i, this means that the customers in $X^i \cap \mathcal{A}^i \cap \mathcal{R}^*$ will be served with total holding costs at most $(\Delta/\delta + 1)R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*)$. The remaining customers left to be served are those in $Y^i \cap \mathcal{R}^*$. Consider customer $k \in Y^i \cap \mathcal{R}^*$. By construction of T^i, it follows that there exists a $t \in T^i$ such that $t \leq a_k \leq t + r^i/h^i \leq a_k + r^i/h^i$. In addition, from the definition of a_k and Lemma 9, it follows that $t_k \in [a_k, a_k + r^i/h^i]$. By the property assumed above, there exists an order that includes item type i within Δ before t. The total holding cost from that order to t, t to a_k, and a_k to t_k is at most $(\Delta + 2r^i/h^i)h^i$ per unit.

It is now sufficient to ensure that indeed for each $t \in T^i$, there exists an order of type i within Δ time periods earlier than t. To achieve this, extra item orders will be added to the construction. From Lemma 10, it then follows that the setup interval corresponding to t intersected some optimal setup interval starting at s_j. (If there is a choice of intersections, choose s_j that contains an item order of type i if one exists.) From Lemma 9, it follows that $s_j - \Delta \leq t \leq s_j + \Delta$. We now consider two cases and show how to enforce the property in each case.

Case 1: There is a type i customer in $\mathcal{A}^*$ with due date in $[t, t+r^i/h^i]$. By construction, this implies that there are type i orders at s_j and $s_j - \Delta$, respectively.

Case 2: There is no type i customer in $\mathcal{A}^*$ with due date in $[t, t + r^i/h^i]$. If $s_j - \Delta \leq t < s_j$, then we place an *extra* item order of type i at the joint order located at time $s_j - \Delta$. Otherwise, $s_j \leq t \leq s_j + \Delta$ and we place the extra order of type i at s_j. Since t corresponds to a setup interval from a stable pair solution, it follows from (2.1) that there exists a set of customers with due dates in $[t, t + r^i/h^i]$

whose rejection costs are greater than K_i. Under the case assumption, the type i demands with due dates in $[t, t + r^i/h^i]$ are all in $\mathcal{R}^*$. Furthermore, all customers with due dates in $[t, t + r^i/h^i]$ are in X^i. Thus, the extra item orders have cost at most $R(X^i \cap \mathcal{R}^*)$. Each customer in $X^i \cap \mathcal{R}^*$ can be used at most once to pay for an extra item order by the spacing of the times we enforced in the construction of T^i.

The total cost incurred by the construction is $K^* + H^* + K^* + \sum_{i=1}^{M}(R(X^i \cap \mathcal{R}^*) + (\Delta/\delta + 1)R(X^i \cap \mathcal{A}^i \cap \mathcal{R}^*) + (\Delta/\delta + 2)R(Y^i \cap \mathcal{R}^*)) \leq 2P(\mathcal{A}^*) + R(\mathcal{R}^*) + (\Delta/\delta + 1)R(\mathcal{A} \cap \mathcal{R}^*)$. Combining Theorems 1 and 2 with this bound completes the proof. $\square$

Proof of Lemma 6. The proof is analogous to the proof of Lemma 5. Let (Q_k, T_k) be the stable pair that the StablePair Algorithm used to accept customer k. Let t be the order that serves customer k in the JR solution of $P(Q_k, T_k)$. Define $Q = \{t\}$, T' to be all the customers served by the order at t, and $\mathcal{I}'$ to be the set of items ordered at t in the solution of $P(Q_k, T_k)$. Note that (Q, T') must be stable pair with $k \in T'$. Define T^i, $\mathcal{I}$, and T according to Properties 1, 2, and 3. Note that $T' \subseteq T$ by Lemma 9 and Property 2. Since $R(T') \geq K_0 + \sum_{i \in \mathcal{I}'} K_i + \sum_{j \in T'} h(t_j - t)d_j$ by (2.1), then it is easy to see that $R(T) \geq K_0 + \sum_{i \in \mathcal{I}} K_i + \sum_{j \in T} h(t_j - t)d_j$, satisfying Property 4. Therefore, (Q, T) is a stable pair with $k \in T$ that satisfies all four properties. The other direction is trivial. $\square$

Proof of Lemma 13. If there is a customer paying more than r, then the solution $\mathrm{OPT}(Q, T)$ can be improved by rejecting that customer and keeping everything else the same. This would contradict the stability of (Q, T). $\square$

Proof of Lemma 14. Assume that $S_k \cap \mathcal{A}^* = \emptyset$. This means that the solution to $\mathrm{OPT}(U)$ rejected the entire set of customers in S_k. However, by stability of (Q_k, T_k) and (2.1) it follows that $P(\{q_k\}, S_k) \leq R(S_k)$. Therefore, adding the customers in S_k to $\mathcal{A}^*$ will not increase the total cost. This is a contradiction since we chose $\mathcal{A}^*$ to be a maximal solution. $\square$

Proof of Lemma 16. As in Lemma 4, we can show that $\alpha R(\mathcal{R}_\alpha \cap \mathcal{A}^*) \leq P(\mathcal{R}_\alpha \cap \mathcal{A}^*)$. Adding $\alpha R(\mathcal{R}_\alpha \cap \mathcal{R}^*)$ and dividing by α completes the proof. $\square$

Proof of Lemma 17. We use the same notation defined in Section 2.7.5. Lemma 13 now holds for αr. Lemma 14 still holds because the rejection costs are scaled down, so (q_k, S_k) must be a stable pair under the original rejection costs as well.

Similar to Lemma 15, we will construct a solution with bounded costs. Specifically, open all of the facilities in the production plan for $P(\mathcal{A}^*)$ and serve each customer in $\mathcal{A}_\alpha$ by the nearest facility to its location. Consider customer $k \in \mathcal{A}_\alpha \cap \mathcal{R}^*$. By the stability of (q_k, S_k) under the scaling and Lemma 13, it follows that $c(k, q_k)$ and $c(q_k, l)$ are both at most αr. It also follows that $c(l, q_l^*) \leq r$ by stability of the optimal solution and using Lemma 13 with the unscaled costs. Thus, $c(k, q_l^*) \leq (2\alpha + 1)r$, which completes the proof. □

Proof of Lemma 7. Assume k was accepted by StablePair, and let j be the facility that served k in the corresponding production plan. Now define Q and T as described in the lemma. The rest of the proof is almost identical to the proof of Lemma 5 and thus we omit it. □

A.3 Figures

Conservative Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1	1	1	1	1	1
5	1.51	1.50	1.86	1.45	1.40	1.11
10	1.52	1.50	2.29	1.24	1.23	1.06

Table A.1: For each value of r, we do 100 experiments, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

More Demands Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1.40	1.40	1.75	1.21	1.21	1.25
5	1.73	1.73	2.64	1.26	1.23	1.04
10	1.73	1.73	2.5	1.11	1.05	1

Table A.2: For each value of r, we do 100 experiments, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

Large Orders First Scenario

	Maximum performance ratio			Final performance ratio		
r	Copycat	StablePair	StablePair(2)	Copycat	StablePair	StablePair(2)
1	1.31	1.31	1.56	1.19	1.19	1.18
5	1.26	1.24	1.21	1.15	1.06	1
10	1.11	1.06	1.03	1	1	1

Table A.3: For each value of r, we do 100 experiments, each with $N = 500$ customer arrivals ($K = 100$ and $h = 1$). We report the maximum performance ratio observed over all experiments and customer arrivals. We also report the average final performance ratio after the final customer has arrived. StablePair(2) denotes the StablePair Algorithm with a scaling factor of 2.

Appendix B

Lower Bound for Steiner Tree with Online Customer Selection

In this section, we derive a lower bound on the competitive ratio for the Steiner tree problem with online customer selection. The proof is similar in spirit to Imase and Waxman [46] who provided a lower bound for the online Steiner problem without customer selection. Note that this proof also holds for the Steiner forest, single-commidity rent-or-buy problem, and multi-commodity rent-or-buy problems with online customer selection since they are all generalizations.

Theorem 18. *The competitive ratio for any deterministic or randomized algorithm for the Steiner tree problem with online customer selection is* $\Omega\left(\sqrt{\log N}\right)$.

Proof. We begin by constructing the same graph as in Imase and Waxman [46]. The procedure to build graph G_k is as follows. Begin with two nodes, v_0 (bottom) and v_1 (top) connected by an edge of length 1, and call this graph G_0. Now to create graph G_k, take every edge in G_{k-1} and replace it by 2 paths, each consists of 2 new edges and 1 new node. The lengths of the 4 new edges will each be half of the length of the original edge, so their length is 2^k. The new nodes are called level k nodes, and each pair of new nodes created are called k-adjacent to each other. See Figure B-1, B-2, B-3, and B-4 for an example of G_3 (nodes are labeled according to their level, each edge is of length 1/8).

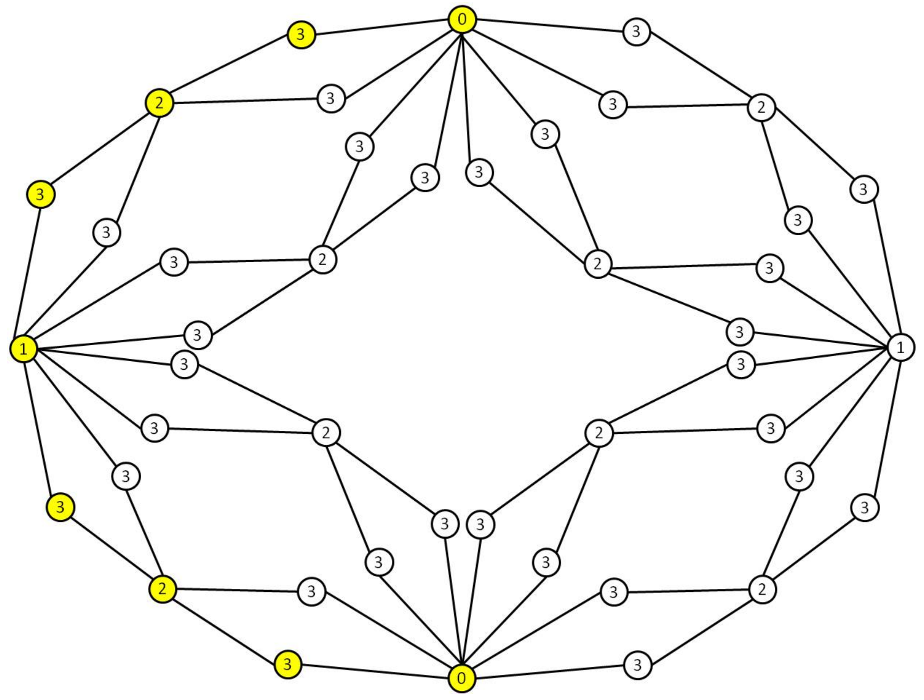

Figure B-1: In this figure, $V^{\hat{\mathcal{A}}} = \{0, 1, 2, 3\}$.

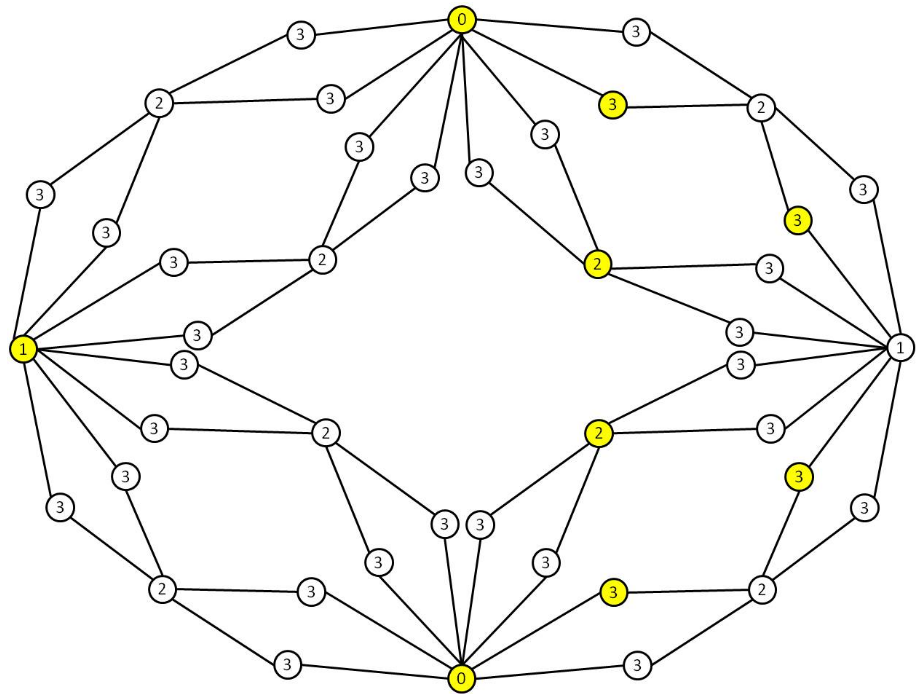

Figure B-2: In this figure, $V^{\hat{\mathcal{A}}} = \{0\}$.

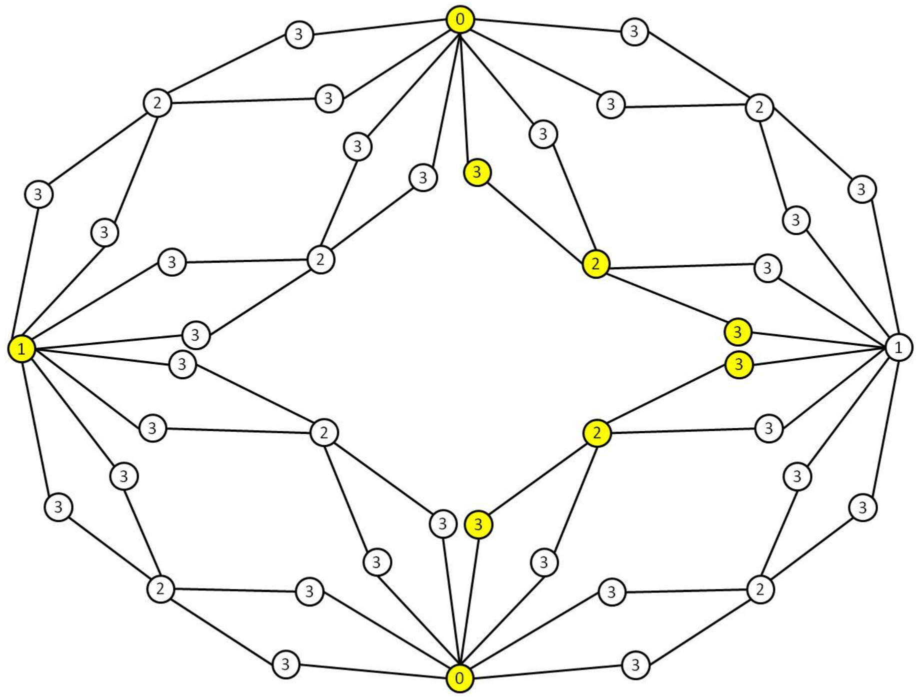

Figure B-3: In this figure, $V^{\hat{\mathcal{A}}} = \{0, 2, 3\}$.

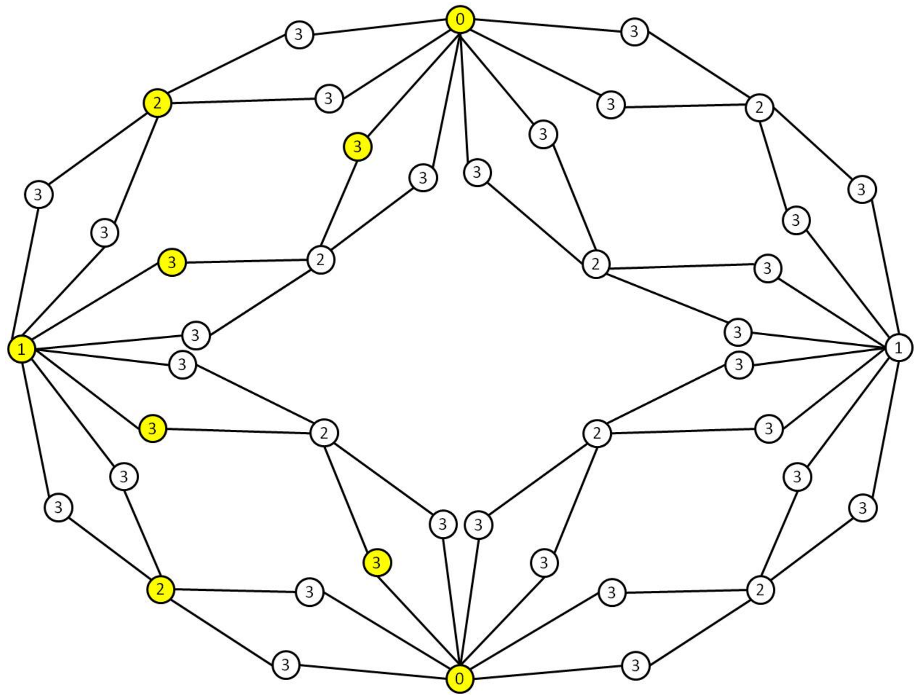

Figure B-4: In this figure, $V^{\hat{\mathcal{A}}} = \{0, 1, 3\}$.

The procedure below demonstrates how we will generate customers. We begin by having 1 customer at level 1, 2 customers at level 2, 4 customers at level 3, and so on until we have 2^{k-1} customers at level k. The rejection cost of a level i customer is $r_i = 1/(2^{i-1}\sqrt{k})$. $V^{\hat{\mathcal{A}}}$ and $V^{\hat{\mathcal{R}}}$ denote the node levels that are accepted and rejected, respectively, by the feasible solution we will consider. See Figures B-1-B-4 for examples on G_3 of the customer sequence.

1. Let $V := v_0$, $i := 0$, $V^{\hat{\mathcal{A}}} = \{0\}$ and $V^{\hat{\mathcal{R}}} = \emptyset$. Let the first two customers be v_0 and v_1, each with infinite rejection cost (must accept).
2. Let $i = i + 1$. Set V_i :=left i-level nodes of V. Generate 2^{i-1} customers, one at each node in V_i, each with rejection cost of $r_i = 1/(2^{i-1}\sqrt{k})$.
3. With probability $1-1/\sqrt{k}$, set $V := V_i$ and let $V^{\hat{\mathcal{A}}} = \{i\} \cup V^{\hat{\mathcal{A}}}$. Otherwise, with probability $1/\sqrt{k}$, set V :=i-adjacent nodes of V_i and $V^{\hat{\mathcal{R}}} = \{i\} \cup V^{\hat{\mathcal{R}}}$.
4. If $i < k$, go to Step 2. Else, Stop.

By construction, it's clear that all the customer nodes with levels in $V^{\hat{\mathcal{A}}}$ can be connected by a single path of length 1. The cost of rejecting all the nodes at any given level is $2^{i-1}r_i = 2^{i-1}(1/(2^{i-1}\sqrt{k})) = 1/\sqrt{k}$. The expected number of levels that are rejected is $\mathbb{E}[|V^{\hat{\mathcal{R}}}|]$, which we can compute to be $(1/\sqrt{k})k = \sqrt{k}$ since the probability that any level is in $V^{\hat{\mathcal{R}}}$ is $1/\sqrt{k}$. The total cost of the feasible solution implied by $\hat{\mathcal{A}}$ and $\hat{\mathcal{R}}$ is then $1 + \mathbb{E}[|V^{\hat{\mathcal{R}}}|]/\sqrt{k} = 2$.

Now let $V^{\mathcal{A}}$ be the levels where an online algorithm accepted at least half of the customers, and let $V^{\mathcal{R}}$ be the levels where an online algorithm rejected at least half of the customers. By construction, in expectation $1/\sqrt{k}$ of the levels in $V^{\mathcal{A}}$ will be in $V^{\hat{\mathcal{R}}}$ due to the random customer distribution. By construction, nodes in level $i \in V^{\hat{\mathcal{R}}}$ can only be connected to any other nodes if they each pay a cost of at least $1/2^i$. Therefore, the cost of connecting half the nodes in any level of $V^{\hat{\mathcal{R}}}$ is $(1/2)(2^{i-1})(1/2^i) = 1/4$.

Since on average $1/\sqrt{k}$ of levels in $V^{\mathcal{A}}$ are in in $V^{\hat{\mathcal{R}}}$, then the cost of serving nodes in levels of $V^{\mathcal{A}}$ is at least $\mathbb{E}[|V^{\mathcal{A}}|]/4\sqrt{k}$. The cost of rejecting half of the nodes in levels of $|V^{\mathcal{R}}|$ is computed similarly as before, and is $\mathbb{E}[|V^{\mathcal{R}}|]/2\sqrt{k}$. Therefore the total cost of the online algorithm is at least $\mathbb{E}[|V^{\mathcal{A}}|]/4\sqrt{k} + \mathbb{E}[|V^{\mathcal{R}}|]/4\sqrt{k} = \sqrt{k}/4$.

Thus, the cost ratio between the online and feasible solution is $O(\sqrt{k})$. So if we have $2 + 2^0 + 2^1 + \ldots + 2^k = N$ total customers, then k is at most $\Theta(\log N)$ which completes the result. $\square$

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Chapter 3 Appendix

C.1 Additional Proofs

C.1.1 Proof of Theorem 1 - Optimal Fulfillment Policy

Here we prove Theorem 1 under a more general setting, where the retailer is selling N transparent products and an opaque product. All of the transparent products are assumed to have the same demand rate. (Symbols $\mathbb{W}$ and $\mathbb{W}^+$ denote the sets of integers and strictly positive integers, respectively. The set of integers from 1 to n is denoted by $[n]$.)

For each inventory position $\mathbf{x} \in \mathbb{W}^N$, where x_i denote the number of inventories of product i on hand, let $J(\mathbf{x})$ denote the expected number of products sold before the next replenishment under the optimal inventory control policy. Then, by definition, $J(\mathbf{x}) = 0$ if $x_i = 0$, for some $i \in [N]$. If $\mathbf{x} \in (\mathbb{W}^+)^N$, then

$$J(\mathbf{x}) = 1 + \frac{1-q}{N} \sum_{i=1}^{N} J(\mathbf{x} - \mathbf{e}^i) + q \max_{i \in [N]} J(\mathbf{x} - \mathbf{e}^i), \tag{C.1}$$

where $\mathbf{e}^i$ is the i-th standard basis vector in $\mathcal{R}^N$, i.e., $e^i_i = 1$, $e^i_j = 0$ for all $j \in [N], j \neq i$. Therefore, the optimal inventory policy can be formulated as a discrete-time stochastic dynamic program, where $J(\cdot)$ is the value function. Next, we establish two properties of the value function which will allow us to the establish the optimal inventory policy.

Lemma 33. *$J(.)$ is symmetric. That is, for any $\mathbf{x} \in (\mathbb{W}^+)^N$, and any σ that is a permutation of $[N]$, $J(\mathbf{x}) = J(\mathbf{x}^\sigma)$, where $\mathbf{x}^\sigma = [x_{\sigma(1)}, x_{\sigma(2)}, ..., x_{\sigma(N)}]$.*

Proof. For any $\mathbf{x} \in \mathbb{W}^N \setminus (\mathbb{W}^+)^N$, $J(\mathbf{x}) = 0$ and is hence symmetric in $\mathbb{W}^N \setminus (\mathbb{W}^+)^N$. For any $\mathbf{x} \in (\mathbb{W}^+)^N$, since the recursive definition of $J(\mathbf{x})$ is symmetric, we must have that $J(.)$ is symmetric under $(\mathbb{W}^+)^N$. □

Lemma 34. *For any $\mathbf{x} \in (\mathbb{W}^+)^N$ and for any i, j where $x_i \leq x_j$, $i \neq j$, $J(\mathbf{x}) > J(\mathbf{x} - \mathbf{e}^i + \mathbf{e}^j)$.*

Proof. We will prove Lemma 34 by induction on $\sum_{k=1}^N x_k$. If $\sum_{k=1}^N x_k = N$, we must have $x_k = 1$ for all $k \in [N]$ and by definition, $J(\mathbf{x}) = 1$, while for any $i \neq j$, $J(\mathbf{x} - \mathbf{e}^i + \mathbf{e}^j) = 0$.

Suppose Lemma 34 is true for any $\mathbf{x}$ where $\sum_{k=1}^N x_k \leq t$. Consider an arbitrary vector $\mathbf{x}$ where $\sum_{k=1}^N x_k = t + 1$, and an pair of distinct integers i, j where $x_i \leq x_j$. For notational simplicity, we let $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{e}^i + \mathbf{e}^j$.

Now, the difference between $J(\mathbf{x}) - J(\hat{\mathbf{x}})$ can be expressed as

$$J(\mathbf{x}) - J(\hat{\mathbf{x}}) = \frac{1-q}{N} \sum_{k=1}^N (J(\mathbf{x} - \mathbf{e}^k) - J(\hat{\mathbf{x}} - \mathbf{e}^k)) + q(\max_{k \in [N]} J(\mathbf{x} - \mathbf{e}^k) - \max_{k \in [N]} J(\hat{\mathbf{x}} - \mathbf{e}^k)). \tag{C.2}$$

For any $k \in [N]$, if $k \neq j$, because $x_i - e_i^k \leq x_i \leq x_j = x_j - e_j^k$, and $\sum_{l=1}^N (x_l - e_l^k) = t$, by induction hypothesis, we have

$$J(\mathbf{x} - \mathbf{e}^k) > J(\hat{\mathbf{x}} - \mathbf{e}^k), \forall k \in [N] \setminus \{j\}. \tag{C.3}$$

For $k = j$, if $x_i = x_j$, then $\hat{\mathbf{x}} - \mathbf{e}^j = \mathbf{x} - \mathbf{e}^i$ is a permutation of $\mathbf{x} - \mathbf{e}^j$, and by Lemma 33, $J(\mathbf{x} - \mathbf{e}^j) = J(\hat{\mathbf{x}} - \mathbf{e}^j)$; if $x_i < x_j$, then $x_i - e_i^j = x_i \leq x_j - 1 = x_j - e_j^j$ and by induction hypothesis, $J(\mathbf{x} - \mathbf{e}^j) \geq J(\hat{\mathbf{x}} - \mathbf{e}^j)$. Therefore, we have

$$J(\mathbf{x} - \mathbf{e}^j) \geq J(\hat{\mathbf{x}} - \mathbf{e}^j). \tag{C.4}$$

Let k^* be such that $J(\hat{\mathbf{x}} - \mathbf{e}^{k^*}) = \max_{k \in [N]} J(\hat{\mathbf{x}} - \mathbf{e}^k)$. Then by Inequalities (C.3) and (C.4), we have that $J(\mathbf{x} - \mathbf{e}^{k^*}) \geq J(\hat{\mathbf{x}} - \mathbf{e}^{k^*})$. This implies

$$\max_{k \in [N]} J(\mathbf{x} - \mathbf{e}^k) \geq J(\mathbf{x} - \mathbf{e}^{k^*}) \geq \max_{k \in [N]} J(\hat{\mathbf{x}} - \mathbf{e}^k). \tag{C.5}$$

Combining Inequalities (C.3)-(C.5) with Equation (C.2), we obtain that $J(\mathbf{x}) - J(\hat{\mathbf{x}}) > 0$. $\square$

Note that if $x_i > x_i$, we must have $x_i - 1 \geq x_j$, and by Lemma 34, we have $J(\mathbf{x} - \mathbf{e}^i) > J(\mathbf{x} - \mathbf{e}^j)$. This in turn implies that $k' \in \arg\max_{k \in [N]} J(\mathbf{x} - \mathbf{e}^k)$ if and only if $k' \in \arg\max_{k \in [N]} x_k$. Therefore, the optimal inventory policy when an opaque customer arrives is always to assign the customer with a product with the highest number of inventories, which proves Theorem 1.

C.1.2 Proof of Theorem 2

Proof. For part (a), observe that we can model the replenishments according to a renewal process. The length of each renewal period is the sum of $R(q, c)$ interarrivals, each with mean time $1/\lambda$. Since $R(q, c)$ is a random variable, then Wald's equality implies that the time between orders is exactly $\mathbb{E}[R(q, c)]/\lambda$. (Note that whether or not the customer chooses an opaque product is independent of his interarrival time, and thus Wald's equality still holds.) Therefore, the ordering cost rate is K divided by the average length of the time between replenishments.

For part (b), the savings is evaluated by looking at (cost of no opaque - cost of opaque level q)/(cost of no opaque).

For part (c), the relative savings is evaluated by looking at (cost of no opaque - cost of opaque level q)/ (cost of no opaque - cost of all opaque).

For part (d), we compute the average inventory in the system and multiply by H. The initial inventory at the beginning of every renewal is $2c$. The ending inventory just before a renewal occurs is $2c - R(q, c) + 1$, and thus the average inventory on hand is $(4c + 1 - \mathbb{E}[R(q, c)])/2$.

For parts (e) and (f), we use the same idea as in parts (b) and (c). $\square$

C.1.3 Proof of Theorem 3

First, we establish a simple lemma that lower bounds the change in the expected number of customers served between consecutive replenishments, $\mathbb{E}[R(q,c)]$, when we increase q from 0 to a strictly positive real number.

Lemma 35. *For any $c > 1$, $q > 0$, $\mathbb{E}[R(q,c)] - \mathbb{E}[R(0,c)] > \mathbb{P}[R(0,c) < 2c-1]q$. Moreover,*

$$\lim_{q\to 0^+} \frac{\mathbb{E}[R(q,c)] - \mathbb{E}[R(0,c)]}{q} > 0.$$

Proof. For any $q > 0$, consider the following opaque fulfillment policy: the retailer randomly selects product to fulfills the opaque purchase if the assignment does not trigger a replenishment, and fulfills the opaque purchase with the product of the highest inventory otherwise. We call this the "delayed-opaque" policy because it does not take advantage of opaque purchases unless it would result in an order. Under the "delayed-opaque" policy, let the number of customers served between consecutive replenishments be $R'(q,c)$. By Theorem 1, we must have that $\mathbb{E}[R(q,c)] \geq \mathbb{E}[R'(q,c)]$.

Observe that under the "delayed-opaque" policy, the retailer's inventory position behaves exactly the same as the traditional selling strategy. When the retailer is forced to replenish under traditional selling, the "delayed-opaque" policy would prolong the replenishment by at least one customer if the last customer buys a opaque product, and the retailer has more than 1 unit of inventory for one of its product. Therefore, we have

$$\begin{aligned}
&\mathbb{E}[R'(q,c)] = \\
&\mathbb{P}[R(0,c) = 2c-1](2c-1) + \mathbb{P}[R(0,c) < 2c-1]\mathbb{E}[R'(q,c)|R(0,c) < 2c-1] \\
&\geq \mathbb{P}[R(0,c) = 2c-1](2c-1) + \mathbb{P}[R(0,c) < 2c-1](\mathbb{E}[R(0,c)|R(0,c) < 2c-1] + q) \\
&= \mathbb{E}[R(0,c)] + \mathbb{P}[R(0,c) < 2c-1]q.
\end{aligned}$$

Note that for a fixed c, the expected value of $R(q,c)$, $\mathbb{E}[R(q,c)]$ can expressed as a polynomial of q. Therefore, the partial derivative of $\mathbb{E}[R(q,c)]$ at q always exists.

And because $\mathbb{P}[R(0,c) < 2c-1] > 0$, we have

$$\lim_{q\to 0^+} \frac{\mathbb{E}[R(q,c)] - \mathbb{E}[R(0,c)]}{q} \geq \lim_{q\to 0^+} \frac{\mathbb{P}[R(0,c) < 2c-1]q}{q} > 0.$$

$\square$

Proof of Theorem 3. For the part(a), by Lemma 35, $q(0) = 0$, and the condition that $\lim_{\delta\to 0^+} \frac{q(\delta)-q(0)}{\delta}$ exists and is strictly positive, we have that

$$\lim_{\delta\to 0^+} \frac{\mathbb{E}[R(q(\delta),c)] - \mathbb{E}[R(0,c)]}{\delta} \geq \lim_{\delta\to 0^+} \frac{\mathbb{P}[R(0,c) < 2c-1]q(\delta)}{\delta} > 0. \tag{C.6}$$

By definition

$$\begin{aligned}\Pi(\delta) &= \lambda(\delta)(1 - q(\delta)\delta - \frac{K}{\mathbb{E}[R(q(\delta),c)]} - \frac{(4c - \mathbb{E}[R(q(\delta),c)] + 1)H}{2\lambda(\delta)}) \\ &\geq \lambda(0)(1 - q(\delta)\delta - \frac{K}{\mathbb{E}[R(q(\delta),c)]} - \frac{(4c - \mathbb{E}[R(q(\delta),c)] + 1)H}{2\lambda(0)}).\end{aligned} \tag{C.7}$$

Define $\Pi_1(\delta) := 1 - q(\delta)\delta$, $\Pi_2(\delta) := \frac{K}{\mathbb{E}[R(q(\delta),c)]}$, and $\Pi_3(\delta) := \frac{(4c-\mathbb{E}[R(q(\delta),c)]+1)H}{2\lambda(0)}$. Then we have

$$\Pi(0) = \lambda(0)(\Pi_1(0) - \Pi_2(0) - \Pi_3(0)).$$

Combine this with Equation C.7, we have

$$\Pi(\delta) - \Pi(0) \geq \lambda(0)(\Pi_1(\delta) - \Pi_1(0) - (\Pi_2(\delta) - \Pi_2(0)) - (\Pi_3(\delta) - \Pi_3(0))).$$

Therefore, the right derivative of $\Pi(\delta)$ at 0 is lower-bounded by

$$\lambda(0)(\lim_{\delta\to 0^+} \frac{\Pi_1(\delta) - \Pi_1(0)}{\delta} - \lim_{\delta\to 0^+} \frac{\Pi_2(\delta) - \Pi_2(0)}{\delta} - \lim_{\delta\to 0^+} \frac{\Pi_3(\delta) - \Pi_3(0)}{\delta}).$$

Now,

$$\lim_{\delta\to 0^+} \frac{\Pi_1(\delta) - \Pi_1(0)}{\delta} = \lim_{\delta\to 0^+} \frac{1 - q(\delta)\delta - 1 - 0}{\delta} = \lim_{\delta\to 0^+} q(\delta) = 0, \tag{C.8}$$

$$\lim_{\delta\to 0^+}\frac{\Pi_2(\delta)-\Pi_2(0)}{\delta} = \lim_{\delta\to 0^+}\frac{K\mathbb{E}[R(0,c)]-K\mathbb{E}[R(q(\delta),c)]}{\delta\mathbb{E}[R(q(\delta),c)]\mathbb{E}[R(0,c)]}$$
$$= \frac{K}{\mathbb{E}[R(0,c)]^2}\lim_{\delta\to 0^+}\frac{\mathbb{E}[R(0,c)]-\mathbb{E}[R(q(\delta),c)]}{\delta} < 0, \qquad \text{(C.9)}$$

$$\text{and } \lim_{\delta\to 0^+}\frac{\Pi_3(\delta)-\Pi_3(0)}{\delta} = \lim_{\delta\to 0^+}\frac{H(\mathbb{E}[R(0,c)]-\mathbb{E}[R(q(\delta),c)])}{\lambda(0)\delta} < 0, \qquad \text{(C.10)}$$

where the Inequalities in Equations C.9 and C.10 hold because of Equation C.6.

Combine Equations (C.8-C.10), we obtain

$$\lambda(0)(\lim_{\delta\to 0^+}\frac{\Pi_1(\delta)-\Pi_1(0)}{\delta} - \lim_{\delta\to 0^+}\frac{\Pi_2(\delta)-\Pi_2(0)}{\delta} - \lim_{\delta\to 0^+}\frac{\Pi_3(\delta)-\Pi_3(0)}{\delta}) > 0,$$

which implies that the right derivative of $\Pi(\delta)$ is strictly positive.

With the formula above, we immediately get that there exists some δ_0 where $\frac{\Pi(\delta_0)-\Pi(0)}{\delta} > 0$, and therefore $\Pi(\delta_0) - \Pi(0)$.

For the second part of Theorem 3, because $q(0) = 0$ and $\lim_{\delta\to 0^+}\frac{q(\delta)-q(0)}{\delta}$ exists, $\lim_{\delta\to 0^+} q(\delta) = q(0) = 0$. Therefore, we get that the right derivative of the revenue rate is

$$= \lim_{\delta\to 0^+}\frac{\lambda(\delta)(1-q(\delta)\delta)-\lambda(0)}{\delta} = \lim_{\delta\to 0^+}\frac{\lambda(\delta)-\lambda(0)}{\delta} - \lim_{\delta\to 0^+}\lambda(\delta)q(\delta)$$
$$= \lim_{\delta\to 0^+}\frac{\lambda(\delta)-\lambda(0)}{\delta}$$
$$> 0.$$

Then there must exist some $\delta_0 > 0$ where $\frac{\lambda(\delta)(1-q(\delta)\delta)-\lambda(0)}{\delta}$ and $\frac{\Pi(\delta_0)-\Pi(0)}{\delta_0} > 0$. This implies that $\Pi_1(\delta_0)) > \Pi_1(0)$ and $\Pi(\delta_0) > \Pi(0)$. □

C.1.4 Proof of Lemma 4

Proof. Observe that $n_t^1(0)-n_t^2(0) = \sum_{i=1}^t X_i$, where X_i are IID random variables that equal to 1 and -1 with probability 0.5 (and therefore mean of 0 and standard deviation of 1). Also, note that for any integer x that is between c and $2c-1$, if $R(0,c) = x$, then

$M_x(0) = |n_x^1(0) - n_x^2(0)| = c - (x-c) = 2c - x$ since we know we sold exactly c of one product for a replenishment to occur. Because $M_{2c}(0) = |n_x^1(0) - n_x^2(0) + \sum_{i=x+1}^{2c} X_i|$ and $\mathbb{P}[|\sum_{i=x+1}^{2c} X_i| \le 2c - x] = 1$, we thus know that

$$\mathbb{E}[M_{2c}(0)|M_x(0) = 2c-x] = \mathbb{E}[|2c - x + \sum_{i=x+1}^{2c} X_i|] = \mathbb{E}[2c - x + \sum_{i=x+1}^{2c} X_i] = 2c - x.$$

Therefore, we know that $\mathbb{E}[M_x(0)|R(0,c) = x] = \mathbb{E}[M_{2c}(0)|R(0,c) = x]$ for any integer x that is between c and $2c-1$. Now since $\mathbb{P}[c \le R(0,c) \le 2c-1] = 1$ and $R(1,c) = 2c-1$, we have that

$$\mathbb{E}[M_{2c}(0)] = \mathbb{E}[M_{R(0,c)}(0)] = 2c - \mathbb{E}[R(0,c)] = 1 + \mathbb{E}[R(1,c) - R(0,c)]. \qquad \text{(C.11)}$$

By Central Limit Theorem (CLT), $\frac{\sum_{i=1}^{2c} X_i}{\sqrt{2c}}$ converges in distribution $N(0,1)$, the normal distribution with zero mean and standard deviation of 1. Because

$$\mathbb{E}[(\frac{\sum_{i=1}^{2c} X_i}{\sqrt{2c}})^2] = \frac{\sum_{i=1}^{2c} \mathbb{E}[X_i^2] + \sum_{1 \le i < j \le 2c} \mathbb{E}[X_i X_j]}{2c} = \mathbb{E}[X_1^2] = 1,$$

it implies (see Example 5 on pg. 351 of [41]) that the sequence of random variables, $\{\frac{\sum_{i=1}^{2c} X_i}{\sqrt{2c}} : c \ge 1\}$ is uniformly integrable. By Skorokhod's representation theorem, we can find random variables such that Y_c converges almost surely to Y, Y_c has the same distribution as $\frac{\sum_{i=1}^{2c} X_i}{\sqrt{2c}}$ for any positive integer c, and Y has the same distribution as $N(0,1)$. Because $\{Y_c : c \ge 1\}$ is uniformly integrable, by Theorem 3 on pg. 351 of [41], we have

$$\lim_{c\to\infty} \mathbb{E}[|\frac{\sum_{i=1}^{2c} X_i}{\sqrt{2c}}|] = \lim_{c\to\infty} \mathbb{E}[|Y_c|] = \mathbb{E}[|Y|] = \mathbb{E}[|N(0,1)|] = \sqrt{\frac{2}{\pi}}. \qquad \text{(C.12)}$$

Finally, combining Equations C.11 and C.12, we have

$$\lim_{c\to\infty} \frac{\mathbb{E}[R(1,c) - R(0,c)]}{\sqrt{2c}} = \lim_{c\to\infty} \frac{\mathbb{E}[-1 + M_{2c}(0)]}{\sqrt{2c}} = \lim_{c\to\infty} \frac{-1 + \mathbb{E}[|\sum_{i=1}^{2c} X_i|]}{\sqrt{2c}} = \sqrt{\frac{2}{\pi}}.$$

□

C.1.5 Proof of Lemma 22

Proof. Given $c \leq x \leq 2c-1$, if $M_x(q) \geq 2c-x$, then either $n_x^1 - n_x^2 \geq 2c-x$ or $n_x^2 - n_x^1 \geq 2c-x$. By definition, $n_x^1 + n_x^2 = x$, and thus, we have $2n_x^1 \geq 2c$ or $2n_x^2 \geq 2c$. This implies that a replenishment which implies that $R(q,c) \leq x$.

On other hand, assume $R(q,c) \leq x$ and let $x' = R(q,c)$. Note that by definition of $R(q,c)$, $\max\{n_{x'}^1, n_{x'}^2\} = c$ and $\min\{n_{x'}^1, n_{x'}^2\} = x'-c$. Thus, we have $M_{x'}(q) = 2c-x'$. Now, because $M_{t+s}(q) \geq M_t(q) - s$, then we have that $M_x(q) = M_{x'+(x-x')}(q) \geq M_{x'}(q) - (x-x') = 2c - x' - (x-x') = 2c-x$.

Thus, we have that for any fixed random instance, $R(q,c) \leq x$ if and only if $M_x(q) \geq 2c-x$, and therefore

$$\mathbb{P}[R(q,c) \leq x] = \mathbb{P}[M_x(q) \geq 2c-x].$$

□

C.1.6 Preservation of Stochastic Dominance in Theorem 5

To complete the gap in the proof of Theorem 5 we want to prove that for any two non-negative random variables D, D' with positive probability at only even integers, if D stochastically dominates D', then $X = (N_1 | N_0 \overset{d}{=} D)$ stochastically dominates $Y = (N_1 | N_0 \overset{d}{=} D')$.

Proof. Let $p_{2i} = \mathbb{P}[D = 2i]$, $p'_{2i} = \mathbb{P}[D' = 2i]$ for each non-negative integer i. For each integer $k \geq 2$, we have

$$\mathbb{P}[X \geq 2k] = p_{2k-2}\frac{(1-q)^2}{4} + p_{2k}(1 - \frac{(1+q)^2}{4}) + \sum_{i=k+1}^{\infty} p_{2i}$$
$$\mathbb{P}[Y \geq 2k] = p'_{2k-2}\frac{(1-q)^2}{4} + p'_{2k}(1 - \frac{(1+q)^2}{4}) + \sum_{i=k+1}^{\infty} p'_{2i}.$$

Because $(1 - \frac{(1+q)^2}{4}) - \frac{(1-q)^2}{4} = \frac{2-2q^2}{4} = \frac{1-q^2}{2} \geq 0$, and $1 - (1 - \frac{(1+q)^2}{4}) = \frac{(1+q)^2}{4} \geq 0$ we have

$$\mathbb{P}[X \geq 2k] = \frac{(1-q)^2}{4} \sum_{i=k-1}^{\infty} p_{2i} + (\frac{1-q^2}{2}) \sum_{i=k}^{\infty} p_{2i} + \frac{(1+q)^2}{4} \sum_{i=k+1}^{\infty} p_{2i}$$
$$\mathbb{P}[Y \geq 2k] = \frac{(1-q)^2}{4} \sum_{i=k-1}^{\infty} p'_{2i} + (\frac{1-q^2}{2}) \sum_{i=k}^{\infty} p'_{2i} + \frac{(1+q)^2}{4} \sum_{i=k+1}^{\infty} p'_{2i}.$$

Thus, we have $\mathbb{P}[X \geq 2k] \geq \mathbb{P}[Y \geq 2k]$, as $\sum_{i=t}^{\infty} p_{2i} \geq \sum_{i=t}^{\infty} p'_{2i}$ for any non-negative integer t due to the fact that D stochastically dominates D'.

For $k = 1$, we have

$$\mathbb{P}[X \geq 2] = p_0 \frac{1-q}{2} + p_2(1 - \frac{(1+q)^2}{4}) + \sum_{i=k+1}^{\infty} p_{2i}$$
$$\mathbb{P}[Y \geq 2] = p'_0 \frac{1-q}{2} + p'_2(1 - \frac{(1+q)^2}{4}) + \sum_{i=k+1}^{\infty} p'_{2i}.$$

Because $(1 - \frac{(1+q)^2}{4}) - \frac{1-q}{2} = \frac{1-q^2}{4} \geq 0$, and $1 - (1 - \frac{(1+q)^2}{4}) = \frac{(1+q)^2}{4} \geq 0$ we can apply the similar argument above and have $\mathbb{P}[X \geq 2] \geq \mathbb{P}[Y \geq 2]$.

Finally, by definition, $\mathbb{P}[X \geq 0] = \mathbb{P}[Y \geq 0] = 1$, and therefore we have that $\mathbb{P}[X \geq 2k] \geq \mathbb{P}[Y \geq 2k]$ for all non-negative integer k. Because X and Y are defined to have only positive probabilities on non-negative even integers, we have that X stochastically dominates Y. $\square$

C.2 Computing $\mathbb{E}[N_\infty]$ and $\mathbb{E}[N'_\infty]$ in Theorem 5

Computing $\mathbb{E}[N_\infty]$. Let $\boldsymbol{\pi}$ be the vector such that

$$\pi_0 = \frac{2q}{1+q}, \pi_{2i} = \frac{4q(1-q)^{2i-1}}{(1+q)^{2i+1}}, \pi_{2i-1} = 0, \forall i = 1, 2, ...$$

Suppose $\mathbb{P}[N^* = i] = \pi_i$, note that

$$\sum_{i=1}^{\infty} \pi_{2i} = \frac{4q(1-q)}{(1+q)^3} \sum_{i=0}^{\infty} \frac{(1-q)^{2i}}{(1+q)^{2i}} = \frac{4q(1-q)}{(1+q)^3} \frac{1}{1-(\frac{1-q}{1+q})^2} = \frac{4q(1-q)}{(1+q)^3} \frac{1}{\frac{4q}{(1+q)^2}} = \frac{1-q}{1+q}.$$

and therefore, $\sum_{i=0}^{\infty} \pi_i = \frac{2q}{1+q} + \frac{1-q}{1+q} = 1$. Moreover,

$$\begin{aligned}
\mathbb{P}[N_1 = 0 | N_0 \overset{d}{=} N^*] &= \pi_0 \cdot \frac{1+q}{2} + \pi_2 \cdot \frac{(1+q)^2}{4} = \frac{2q}{1+q}, \\
\mathbb{P}[N_1 = 2i | N_0 \overset{d}{=} N^*] &= \pi_{2i} \frac{1-q^2}{2} + \pi_{2(i-1)} \frac{(1-q)^2}{4} + \pi_{2(i+1)} \frac{(1+q)^2}{4} \\
&= \frac{4q(1-q)^{2i-1}}{(1+q)^{2i+1}}, \forall i \geq 1.
\end{aligned}$$

Therefore, $\boldsymbol{\pi}$ is indeed the steady state vector of N_i and $N_\infty \overset{d}{=} N^*$. Now

$$\begin{aligned}
\mathbb{E}[N^*] = \sum_{i=1}^{\infty} 2i \cdot \pi_{2i} &= \frac{8q(1-q)^{-1}}{(1+q)} \sum_{i=1}^{\infty} i (\frac{(1-q)^2}{(1+q)^2})^i \\
&= \frac{8q}{(1+q)(1-q)} \frac{(1-q)^2}{(1+q)^2} \frac{1}{(1-\frac{(1-q)^2}{(1+q)^2})^2} \\
&= \frac{(1-q)(1+q)}{2q}.
\end{aligned}$$

Computing $\mathbb{E}[N'_\infty]$. Let $\boldsymbol{\pi}$ be the vector such that

$$\pi_{2i+1} = \frac{4q(1-q)^{2i}}{(1+q)^{2i+2}}, \ \pi_{2i} = 0, \ \forall i = 0, 2, ...$$

Suppose $\mathbb{P}[N^* = i] = \pi_i$, note that

$$\begin{aligned}
\sum_{i=0}^{\infty} \pi_{2i+1} = \sum_{i=0}^{\infty} \frac{4q(1-q)^{2i}}{(1+q)^{2i+2}} &= \frac{4q}{(1+q)^2} \frac{1}{1-\frac{(1-q)^2}{(1+q)^2}} = 1 \\
\mathbb{P}[N'_1 = 1 | N'_0 \overset{d}{=} N^*] &= \pi_1 \cdot \frac{(1+q)(3-q)}{4} + \pi_3 \cdot \frac{(1+q)^2}{4} = \frac{4q}{(1+q)^2}, \\
\mathbb{P}[N'_1 = 2i+1 | N'_0 \overset{d}{=} N^*] &= \pi_{2i} \frac{1-q^2}{2} + \pi_{2(i-1)} \frac{(1-q)^2}{4} + \pi_{2(i+1)} \frac{(1+q)^2}{4} \\
&= \frac{4q(1-q)^{2i}}{(1+q)^{2i+2}}, \forall i \geq 1.
\end{aligned}$$

Therefore, $\boldsymbol{\pi}$ is the steady state vector of N_i' and $N_\infty' \overset{d}{=} N^*$. Now we can show that $\mathbb{E}[N^*] = \frac{1+q^2}{2q}$.

C.3 Additional Figures

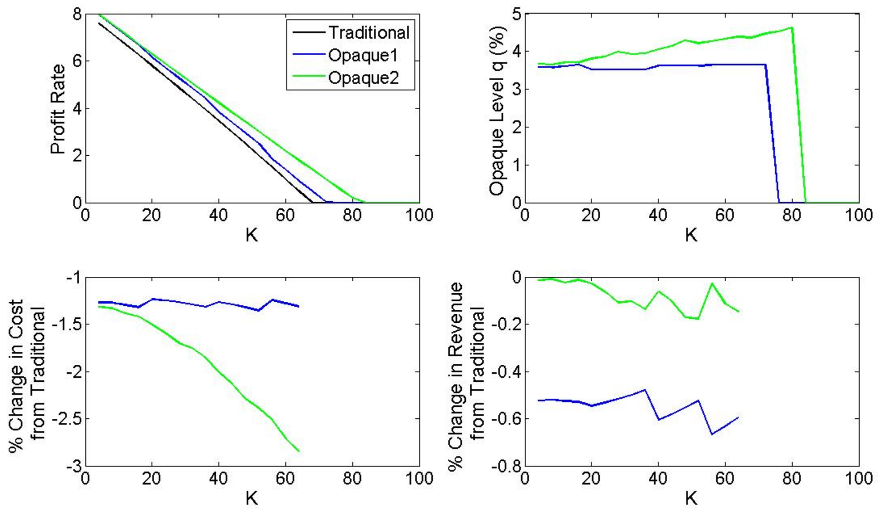

Figure C-1: The black lines correspond to the optimal traditional strategy, the blue lines to the optimal Opaque1 strategy, and the green lines to the optimal Opaque2 strategy.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] E. Arkin, D. Joneja, and R. Roundy. Computational complexity of uncapacitated multi-echelon production planning problems. *Oper. Res. Lett.*, 8(2):61–66, 1989. ISSN 0167-6377.

[2] S. Axsater. Worst case performance for lot sizing heuristics. *EJOR*, 9(4):339–343, 1982. ISSN 0377-2217.

[3] Joseph L Balintfy. On a basic class of multi-item inventory problems. *Management Science*, 10(2):287–297, 1964.

[4] M.O. Ball and M. Queyranne. Toward robust revenue management: Competitive analysis of online booking. *Oper. Res.*, 57(4):950–963, 2009. ISSN 0030-364X.

[5] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 184–193. IEEE, 1996.

[6] L Becchetti, P Korteweg, A Marchetti-Spaccamela, M Skutella, L Stougie, and A Vitaletti. *Latency constrained aggregation in sensor networks.* Springer, 2006.

[7] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. Dynamic Ideas Belmont, 2005.

[8] S. Bhaskaran, K. Ramachandran, and J. Semple. A dynamic inventory model with the right of refusal. *Management Science*, 56(12):2265–2281, 2010.

[9] Sreekumar R Bhaskaran and Vish Krishnan. Effort, revenue, and cost sharing mechanisms for collaborative new product development. *Management Science*, 55(7):1152–1169, 2009.

[10] Marcin Bienkowski, Jaroslaw Byrka, Marek Chrobak, Lukasz Jez, and Jiri Sgall. Better approximation bounds for the joint replenishment problem. *arXiv preprint arXiv:1307.2531*, 2013.

[11] D. Bienstock, M.X. Goemans, D. Simchi-Levi, and D. Williamson. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59 (1):413–420, 1993.

[12] Y. Bleischwitz and F. Schoppmann. New efficiency results for makespan cost sharing. *Information Processing Letters*, 107(2):64–70, 2008.

[13] J. Brenner and G. Schäfer. Cost sharing methods for makespan and completion time scheduling. *STACS 2007*, pages 670–681, 2007.

[14] N. Buchbinder, T. Kimbrelt, R. Levi, K. Makarychev, and M. Sviridenko. Online make-to-order joint replenishment model: primal dual competitive algorithms. *Oper. Res.*, page Forthcoming, 2014.

[15] G.P. Cachon. Supply chain coordination with contracts. *Handbooks in operations research and management science*, 11:229–340, 2003.

[16] R. Caldentey and L. M. Wein. Revenue management of a make-to-stock queue. *Oper. Res.*, 54(5):859–875, 2006.

[17] S. Carr and I. Duenyas. Optimal admission control and sequencing in a make-to-stock/make-to-order production system. *Oper. Res.*, 48(5):709–720, 2000.

[18] Oben Ceryan, Ozge Sahin, and Izak Duenyas. Dynamic pricing of substitutable products in the presence of capacity flexibility. *Manufacturing & Service Operations Management*, 15(1):86–101, 2013.

[19] L.M.A. Chan, A. Muriel, Z.J.M. Shen, D. Simchi-Levi, and C.P. Teo. Effective zero-inventory-ordering policies for the single-warehouse multiretailer problem with piecewise linear cost structures. *Management Science*, pages 1446–1460, 2002.

[20] M. Charikar, S. Khuller, D.M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 642–651. Society for Industrial and Applied Mathematics, 2001.

[21] S. Chawla, T. Roughgarden, and M. Sundararajan. Optimal cost-sharing mechanisms for steiner forest problems. *Internet and Network Economics*, pages 112–123, 2006.

[22] Xin Chen and David Simchi-Levi. Coordinating inventory control and pricing strategies with random demand and fixed ordering cost: The finite horizon case. *Operations Research*, 52(6):887–896, 2004.

[23] M. Cheung, A.N. Elmachtoub, R. Levi, and D.B. Shmoys. The submodular joint replenishment problem. *Mathematical Programming*, 2014.

[24] Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[25] S.E. Deering and D.R. Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems (TOCS)*, 8(2): 85–110, 1990. ISSN 0734-2071.

[26] A.N. Elmachtoub and R. Levi. From cost sharing mechanisms to online selection problems. *Mathematics of Opeartions Research*, 2014.

[27] A.N. Elmachtoub and R. Levi. Supply chain management with online customer selection. *Unpublished Manuscript*, 2014.

[28] A.N. Elmachtoub and Y. Wei. Retailing with opaque products. *Unpublished Manuscript*, 2014.

[29] Scott Fay and Jinhong Xie. Probabilistic goods: A creative way of selling products and services. *Marketing Science*, 27(4):674–690, 2008.

[30] A. Federgruen and M. Tzur. The joint replenishment problem with time-varying costs and demands: Efficient, asymptotic and ε-optimal solutions. *Operations Research*, pages 1067–1086, 1994.

[31] A. Federgruen and Y.S. Zheng. The joint replenishment problem with general joint cost structures. *Operations Research*, 40(2):384–403, 1992. ISSN 0030-364X.

[32] Awi Federgruen and Aliza Heching. Combined pricing and inventory control under uncertainty. *Operations research*, 47(3):454–475, 1999.

[33] Awi Federgruen, Maurice Queyranne, and Yu-Sheng Zheng. Simple power-of-two policies are close to optimal in a general class of production/distribution networks with general joint setup costs. *Mathematics of Operations Research*, 17 (4):951–963, 1992.

[34] D. Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[35] Guillermo Gallego and Robert Phillips. Revenue management of flexible products. *Manufacturing & Service Operations Management*, 6(4):321–337, 2004.

[36] J. Gallien, Y. Le Tallec, and T. Schoenmeyr. A model for make-to-order revenue management. Technical report, Citeseer, 2004.

[37] J. Geunes, H.E. Romeijn, and K. Taaffe. Requirements planning with pricing and order selection flexibility. *Oper. Res.*, 54(2):394, 2006.

[38] J. Geunes, R. Levi, H.E. Romeijn, and D.B. Shmoys. Approximation algorithms for supply chain planning and logistics problems with market choice. *Mathematical Programming*, pages 1–22, 2009. ISSN 0025-5610.

[39] J. Geunes, R. Levi, H.E. Romeijn, and D.B. Shmoys. Approximation algorithms for supply chain planning and logistics problems with market choice. *Mathematical programming*, 130(1):85–106, 2011.

[40] M.X. Goemans and D.P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[41] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Process*. Oxford University Press, New York, 2nd edition, 1992.

[42] A. Gupta, A. Srinivasan, and É. Tardos. Cost-sharing mechanisms for network design. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 139–150, 2004.

[43] S. He, J. Zhang, and S. Zhang. Polymatroid optimization, submodularity, and joint replenishment games. *Operations Research*, 60(1):128–137, 2012.

[44] Yale Herer and Robin Roundy. Heuristics for a one-warehouse multiretailer distribution problem with performance bounds. *Operations Research*, 45(1):102–115, 1997.

[45] Edward Ignall. Optimal continuous review policies for two product inventory systems with joint setup costs. *Management Science*, 15(5):278–283, 1969.

[46] M. Imase and B.M. Waxman. Dynamic steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4:369, 1991.

[47] Patrick Jaillet and Xin Lu. Online traveling salesman problems with rejection options. Technical report, Working paper, Operations Research Center, Massachusetts Institute of Technology, 2013.

[48] K. Jain and V.V. Vazirani. Equitable cost allocations via primal-dual-type algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 313–321. ACM, 2002.

[49] Kinshuk Jerath, Serguei Netessine, and Senthil K Veeraraghavan. Selling to strategic customers: Opaque selling strategies. *Consumer-Driven Demand and Operations Management Models*, pages 253–300, 2009.

[50] Kinshuk Jerath, Serguei Netessine, and Senthil K Veeraraghavan. Revenue management with strategic customers: Last-minute selling and opaque selling. *Management Science*, 56(3):430–448, 2010.

[51] D. Joneja. The joint replenishment problem: new heuristics and worst case performance bounds. *Oper. Res.*, 38(4):711–723, 1990. ISSN 0030-364X.

[52] E.P.C. Kao. A multi-product dynamic lot-size model with individual and joint set-up costs. *Operations Research*, pages 279–289, 1979.

[53] P. Keskinocak, R. Ravi, and S. Tayur. Scheduling and reliable lead-time quotation for orders with availability intervals and lead-time sensitive revenues. *Management Science*, 47(2):264–279, 2001.

[54] Sanjeev Khanna, Joseph Seffi Naor, and Dan Raz. Control message aggregation in group communication protocols. In *Automata, Languages and Programming*, pages 135–146. Springer, 2002.

[55] Sang-Hyun Kim and Serguei Netessine. Collaborative cost reduction and component procurement under information asymmetry. *Management Science*, 59(1): 189–206, 2013.

[56] J. Könemann, S. Leonardi, and G. Schäfer. A group-strategyproof mechanism for steiner forests. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 612–619. Society for Industrial and Applied Mathematics, 2005.

[57] R. Levi, R.O. Roundy, and D.B. Shmoys. Primal-Dual Algorithms for Deterministic Inventory Problems. *Mathematics of Operations Research*, 31(2):267–284, 2006.

[58] R. Levi, R. Roundy, D. Shmoys, and M. Sviridenko. A Constant Approximation Algorithm for the One-Warehouse Multiretailer Problem. *Management Science*, 54(4):763, 2008.

[59] R. Levi, T.L. Magnanti, E.J. Zarybnisky, et al. *Maintenance scheduling for modular systems-models and algorithms.* PhD thesis, Massachusetts Institute of Technology, 2011.

[60] S. Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Automata, Languages and Programming*, pages 77–88, 2011.

[61] S.T. McCormick. Submodular function minimization. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Handbooks in Operations Research and Management Science: Discrete Optimization*, volume 12, pages 321–391. Elsevier, 2006.

[62] A. Meyerson. Online facility location. In *Proceedings of 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2002. ISBN 0769511163.

[63] H. Moulin. Incremental cost sharing: Characterization by coalition strategy-proofness. *Social Choice and Welfare*, 16(2):279–320, 1999.

[64] H. Moulin and S. Shenker. Strategyproof sharing of submodular costs: budget balance versus efficiency. *Economic Theory*, 18(3):511–533, 2001.

[65] M. Nagarajan and G. Sosic. Game-theoretic analysis of cooperation among supply chain agents: Review and extensions. *European Journal of Operational Research*, 187(3):719–745, 2008.

[66] Tim Nonner and Alexander Souza. A 5/3-approximation algorithm for joint replenishment with deadlines. In *Combinatorial Optimization and Applications*, pages 24–35. Springer, 2009.

[67] Tim Nonner and Maxim Sviridenko. An efficient polynomial-time approximation scheme for the joint replenishment problem. In *Integer Programming and Combinatorial Optimization*, pages 314–323. Springer, 2013.

[68] M. Pál and É. Tardos. Group strategy proof mechanisms via primal-dual algorithms. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 584–593. IEEE, 2003.

[69] Nicholas C Petruzzi and Maqbool Dada. Pricing and the newsvendor problem: A review with extensions. *Operations Research*, 47(2):183–194, 1999.

[70] E. Plambeck, S. Kumar, and J. M.l Harrison. A multiclass queue in heavy traffic with throughput time constraints: Asymptotically optimal dynamic controls. *Queueing Systems*, 39(1):23–54, 2001.

[71] J. Qian and D. Williamson. An o (logn)-competitive algorithm for online constrained forest problems. *Automata, Languages and Programming*, pages 37–48, 2011.

[72] M Queyranne. A polynomial-time, submodular extension to roundy's 98% effective heuristic for production/inventory. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1640–1640. IEEE, 1986.

[73] T. Roughgarden and M. Sundararajan. New trade-offs in cost-sharing mechanisms. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 79–88. ACM, 2006. ISBN 1595931341.

[74] T. Roughgarden and M. Sundararajan. Optimal efficiency guarantees for network design mechanisms. *Integer Programming and Combinatorial Optimization*, pages 469–483, 2007.

[75] R. Roundy. 98%-effective integer-ratio lot-sizing for one-warehouse multi-retailer systems. *Management science*, pages 1416–1430, 1985.

[76] Andreas S Schulz and Claudio Telha. Approximation algorithms and hardness results for the joint replenishment problem with constant demands. In *Algorithms–ESA 2011*, pages 628–639. Springer, 2011.

[77] Danny Segev. An approximate dynamic-programming approach to the joint replenishment problem. *Mathematics of Operations Research*, 2013.

[78] David Shmoys and Chaoxu Tong. Private communivation, 2013.

[79] D.B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. In *Proceedings of the twenty-ninth annual ACM Symposium on Theory of Computing*, pages 265–274. ACM, 1997. ISBN 0897918886.

[80] Susan A Slotnick. Order acceptance and scheduling: a taxonomy and review. *European Journal of Operational Research*, 212(1):1–11, 2011.

[81] Jing-Sheng Song and Zhengliang Xue. Demand management and inventory control for substitutable products. *Working paper*, 2007.

[82] G. Stauffer, G. Massonnet, C. Rapine, and J.P. Gayon. A simple and fast 2-approximation algorithm for the one-warehouse multi-retailers problem. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2011.

[83] Z. Svitkina and E. Tardos. Facility location with hierarchical facility costs. *ACM Transactions on Algorithms (TALG)*, 6(2):37, 2010.

[84] C.P. Teo and D. Bertsimas. Multistage lot sizing problems via randomized rounding. *Operations Research*, pages 599–608, 2001.

[85] W. Van den Heuvel and A.P.M. Wagelmans. Worst-case analysis for a general class of online lot-sizing heuristics. *Operations research*, 58(1):59–67, 2010.

[86] W. Van den Heuvel, O.E. Kundakcioglu, J. Geunes, H.E. Romeijn, T.C. Sharkey, and A.P.M. Wagelmans. Integrated market selection and production planning: complexity and solution approaches. *Mathematical programming*, 134(2):395–424, 2012.

[87] A.F. Veinott Jr. Minimum concave-cost solution of leontief substitution models of multi-facility inventory systems. *Operations Research*, pages 262–291, 1969.

[88] S Viswanathan. An algorithm for determining the best lower bound for the stochastic joint replenishment problem. *Operations research*, 55(5):992–996, 2007.

[89] H.M. Wagner and T.M. Whitin. Dynamic Version of the Economic Lot Size Model. *Management Science*, 5(1):89–96, 1958. ISSN 0025-1909.

[90] M.R. Wagner. Fully distribution-free profit maximization: the inventory management case. *Math. of Oper. Res.*, 35(4):728–741, 2010.

[91] D. Xu and R. Yang. A cost-sharing method for an economic lot-sizing game. *Operations Research Letters*, 37(2):107–110, 2009.

[92] G. Xu and J. Xu. An improved approximation algorithm for uncapacitated facility location problem with penalties. *Journal of combinatorial optimization*, 17(4):424–436, 2009.

[93] Y. Xu, A. Bisi, and M. Dada. A periodic-review base-stock inventory system with sales rejection. *Operations research*, 59(3):742–753, 2011.

[94] A. C. Yao. Probabilistic computation: towards a unified measure of complexity. In *IEEE Symposium on Foundations of Computer Science*, pages 222–227, 1977.

[95] W.I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system-a network approach. *Management Science*, pages 506–527, 1969.

[96] Kaijie Zhu and Ulrich W Thonemann. Coordination of pricing and inventory control across products. *Naval Research Logistics (NRL)*, 56(2):175–190, 2009.