CARNEGIE MELLON UNIVERSITY

DOCTORAL THESIS

Models and Methods for Omni-channel Fulfillment

Author:
Jeremy KARP

Supervisor:
Dr. R. RAVI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in

Algorithms, Combinatorics, and Optimization
Tepper School of Business, Carnegie Mellon University

September 13, 2017

Declaration of Authorship

I, Jeremy KARP, declare that this thesis titled, "Models and Methods for Omnichannel Fulfillment" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Carnegie Mellon University

Abstract

Tepper School of Business, Carnegie Mellon University

Doctor of Philosophy

Models and Methods for Omni-channel Fulfillment

by Jeremy KARP

Omni-channel retailing, the combination of online and traditional store channels, has led to the use of traditional stores as fulfillment centers for online orders. A key aspect of omni-channel fulfillment problems is the tradeoff between cancellations of accepted online orders and profits: a riskier fulfillment policy may result in more online sales but also more cancelled orders.

In this dissertation, I will describe two approaches to the fulfillment problem cast generally as a stochastic optimization problem of setting inventory thresholds above which the online channel stays open.

In the more traditional approach, we build a stochastic model of the process leading to order cancellations for a single item so that retailers may find inventory and fulfillment policies that effectively use this information along with shipping costs between various locations. We describe iterative algorithms based on Infinitesimal Perturbation Analysis (IPA) that converge to optimal and locally optimal policies within certain flexible policy classes for the multiple-location version of this model, and show their empirical performance on simulated data based on real data from a high-end North American retailer.

In a more modern approach, we apply techniques from machine learning and discrete optimization to find fulfillment policies that perform well empirically at maximizing revenues subject to a constraint on cancellations across a large portfolio of items. Using the real data mentioned earlier, we build estimators that predict the cancellation probability and other features of incoming online orders. We formulate and solve an optimization problem based on these estimates to get a fulfillment policy in a separate second step. Then we investigate a joint estimation and optimization model based on a neural network to find both the generative parameters for our estimates as well as the policy that maximizes revenue while limiting cancellations. We show how both the separate and joint estimation and optimization models can be used to account for data truncation. The joint methods typically identify policies that more closely track target cancel rates than the separate estimation and optimization methods. For the joint method, we also custom built a neural network layer to efficiently solve the knapsack optimization problem central to our model and demonstrate substantial empirical improvement in running time and scalability over existing methods.

Acknowledgements

First, I must thank my advisor, Ravi, who has taught me so much and gave me the opportunity to write this dissertation. I'm grateful for all the time and effort you have spent in meetings and discussions with me as well as all of your feedback on my paper drafts, write-ups, and various notes written throughout my time at CMU. Thank you for patiently exploring many research topics with me and for pushing me to do my best work.

I would also like to thank Prof. Sridhar Tayur for all your help and advice on this dissertation. Thank you for introducing me to all kinds of useful papers, books, and ideas I would otherwise not know about and for helping me structure the ideas and models in my research.

I am also grateful to Dr. Srinath Sridhar for your collaboration during my thesis research. Thank you for introducing me to many interesting challenges in the retail industry and for helping me turn them into research problems, and thank you for allowing me to use Onera's data to inform and validate my results.

Lastly, I would like to thank my family and friends for motivating me and supporting me throughout my PhD. Thank you to my parents for instilling an interest in learning, research, and academics. Thank you to Andrea for supporting me unconditionally throughout graduate school. Thank you to my peers in ACO/OR at CMU: Alex, Christian, Stelios, Thiago, Yang, Jenny, Sercan, and everyone else who motivated me to learn about a wide range of fascinating topics. And thank you to my friends and family who helped me stay connected to the world outside of operations research and machine learning during graduate school.

Contents

List of Figures

Dedicated to my parents

Chapter 1

IPA Methods for Omni-Channel Fulfillment

1.1 Introduction

Omni-channel retailing, the combination of online and traditional store channels, advocates the use of traditional stores as shipping centers for originating online orders, customer pickup points for online orders, or even as transshipment points for re-balancing stock. Many retailers have begun to ship items ordered online directly from their brick-and-mortar locations. This clicks and bricks business model allows retailers to save money by keeping more of their inventory in retail locations as opposed to building or leasing warehouses. When filling these online orders, the retailer must at some point trade off additional cancelled orders in order to increase revenues by accepting additional online orders. We study a new set of research questions related to acceptance and fulfillment of these online orders in omni-channel retail operations. Our models focus on the acceptance and fulfillment decisions for online channels, taking into consideration the costs of fulfillment including shipping costs when they are filled and the possibility of canceling some of the accepted online orders. We tackle the problem of omni-channel fulfillment from a stochastic inventory theory perspective where inventory is held at physical stores and shared between in-store demand and online demand. Physical retail stores, however, are not designed for online fulfillment, and these inventory pooling arrangements often lead to cancelled orders. A major driver of order cancellations is that the required inventory for an online order can be listed in the retailer's inventory database when the order is placed but then depleted by an in-store sale before the item is picked for shipment. Our analytical models optimize the trade-off between policies that fill many online orders, yielding additional revenue, and the penalties incurred from cancelling online orders if too many are accepted. While the tradeoff is the same as the one studied in Chapters 2 and 3, in this chapter we incorporate the effect of shipping costs into our model and introduce a class of policies that consider the locations of inventory demand while making fulfillment decisions.

1.1.1 Problem Description

This work is focussed on fulfillment problems in omni-channel retailing. At the highest level, the problem we model is determining under which conditions should a retailer accept online orders, and the retailer's objective is to minimize fulfillment costs in expectation. There are n regions in which demand can originate, and in each region is a physical retail store. We use the terms store location and region interchangeably. In this model, the retailer has inventory for a single item spread across the n store locations. There are two streams of demand at each store location, in-store and online demand (online orders are attributed to the closest store to each customer), and the core challenge is that inventory is shared between these demand streams. The model has two stages: in the first stage the online orders arrive in sequence and the retailer must decide whether to accept or reject each order as it arrives. This stage reflects that an online retailer can dynamically set an item to be listed as in or out of stock on its website. The second stage occurs after all online orders have been accepted or rejected, and the retailer must decide how to fulfill all accepted online orders. In-store orders are given first priority as these sales are completed as the online orders arrive, and any remaining inventory not used to fulfill in-store orders may be used to fulfill accepted online orders. Any online orders that cannot be fulfilled with the remaining inventory are cancelled and the retailer pays an associated cost. To be clear, this work is not about planning of inventories. Instead, we are interested in the fulfillment of orders given inventory levels.

1.1.2 Summary of Contributions

1. We formulate an analytical model for omni-channel fulfillment that incorporates uncertainty due to inventory pooling across sales channels as a multi-location two-stage stochastic optimization problem.

2. We introduce Local Threshold and Global Threshold policy classes for the first stage problem and present a sampling-based optimization method to set these policies. Our optimization method uses Infinitesimal Perturbation Analysis to estimate derivatives of the objective function with respect to the threshold policy parameters. These derivative estimates rely on the dual values of a linear program related to the second stage problem.

3. We present empirical results from numerical experiments to provide insights and demonstrate the effectiveness of policies generated by our methods. Through a partnership of with retail analytics firm Onera, we use retail industry data to generate realistic problem instances. We conduct a series of experiments on two-store instances to demonstrate how certain instance attributes lead to strong performance of one class of threshold attributes relative to the other. In particular, we find that Local Thresholds perform especially well in settings where inventory is not well aligned with demand, when inventory levels are

low, when in-store demand levels are low, and when cancel costs are low relative to the item's price. We also use Onera's retail data to formulate realistic full-network problem instances on which we show both Local Thresholds and Global Thresholds achieve a considerable reduction in costs compared to other baseline policies.

1.2 Related Work

Other aspects of omni-channel retailing, such as the costs and benefits of "buy online and pick up in store" policies [20], information sharing [21], inventory optimization [27], and multi-channel price optimization [29, 11] have been studied by the Operations Management community, but this is the first attempt to formulate and study stochastic models of cancellations caused by omni-channel fulfillment. Our analyses use techniques that have been successful in other areas within Operations Management including transshipment problems [31], sensitivity analysis [23], and Sample Average Approximation [42, 41].

The multi-location transshipment problem has been previously formulated and studied [31, 30]. This work presents a stochastic multi-period model, and its main theoretical result is that optimal inventory replenishment policies in this model are "order-up-to S" policies. Research on newsvendor models is also closely related to the models in this section [48]. There are many versions and extensions of the newsvendor model, but a central feature of these models is that the vendor must use their knowledge of a demand distribution to select an order quantity so as to balance penalties for ordering both too many and too few items. The models in this section incorporate some of the complexities considered in the multi-location transshipment problem, but these new models also include order cancellations as a major component that must be accounted for by the retailer. The tradeoff between cancellations and additional sales places this model in a similar space as newsvendor models, but the inclusion of two sources of demand drawing from the same inventory adds a layer of complexity that is new. Extending this model to multiple locations combines the newsvendor-like tradeoff of cancellations and sales with fulfillment decisions in a network.

There has also been work on omni-channel fulfillment with an emphasis on pricing [29]. This work models how customer demand responds to changes in price, allowing the retailer to optimally set clearance prices in all sales channels by solving an integer program. In our work prices are fixed and our focus is on fulfillment with uncertain inventory and stochastic demand.

Sample Average Approximation (SAA) [39, 41] has proved to be a successful technique for obtaining theoretical performance guarantees on related problems. SAA is a Monte Carlo simulation-based method for stochastic optimization where the random inputs are approximated by their empirical distribution, the sample average of observed data points. We apply this approach to our omni-channel optimization problem when demand distributions are unknown but can be sampled. SAA has previously been studied in the context of the newsvendor problem [42], and we use similar techniques to prove convergence bounds on versions of this omni-channel problem. Akcay et al. [2] studied a related inventory problem with unknown demand distributions, representing the unknown demand distribution with Johnson translation systems, a parameterized family of distributions that can closely approximate any probability distribution.

1.3 Omni-Channel Fulfillment Model

One objective of omni-channel fulfillment is to provide customers with an integrated experience across all sales channels while allowing the retailer to share inventory between these channels. Our model allows retailers to decide when and how to accept and fulfill online orders when inventory is shared with physical retail locations. We study a single-period model with n regions of demand, each containing a store. The demand comes in two streams, online and in-store. The regional split between online and in-store demand is assumed to be pre-existing. We consider a single product and assume that online and in-store demand are drawn from fixed probability distributions. The core omni-channel problem is to determine under what inventory conditions should the online store be kept open. A crucial consideration for this decision is that if the online store is kept open despite having a low inventory count the retailer becomes exposed to the risk of needing to cancel some accepted online orders, incurring a cancellation cost.

Conversely, closing the online store when sufficient inventory is available results in lost sales through the online sales channel. Online demands that are satisfied are fulfilled only by shipping the order directly to the customer (no walk-in pickups at local stores). An online channel, if open, can tentatively accept orders as demands arrive; a confirmation or a cancellation notice is sent out at the end of the period. This models the reality that online orders are confirmed and fulfilled typically at the end of each day and also that changes in inventory records are not always updated instantaneously. Demands not immediately accepted are considered lost and the retailer is penalized for the associated lost margin. Orders accepted online are to be considered tentative as they may be canceled at the end of the period if no inventory from physical store is found. At the end of the period, each store location can fulfill online orders from its remaining inventory after fulfilling the day's physical orders. On-line orders that are filled pay the appropriate shipping cost and those that are not fulfilled from any of the physical locations are cancelled. The resulting problem is to minimize fulfillment costs by setting a policy to trigger the opening and closing of the online sales channel at each physical location. This model allows for multiple physical stores with differing starting inventory levels. In these scenarios the retailer must also determine a fulfillment policy to ship online orders to their recipients, potentially by filling online orders from far away stores in the network.

1.3.1 Details

The model contains a network of n store locations. Fixed exogenous inventory $\mathcal{I}_i$ is stored at each location i in this system. At each location i, there are two streams of demand, on-line ($\mathcal{D}_i^O$) and physical ($\mathcal{D}_i^P$), which both draw from the same pool of inventory. Physical demand is fulfilled with higher priority than online demand, and online orders are cancelled if there is not sufficient inventory to fill them. There is a cost, c, associated with canceling an order, and there is also a penalty cost, p, associated with not accepting an order that could have been filled. The goal for the retailer is to set a policy so as to minimize total costs in expectation. The retailer may be given the probability distributions of demand or it may be unknown, and their costs are determined by how many orders are filled and how many are cancelled after these demand distributions are realized. We capture this process through a two-stage stochastic model.

1.3.2 First Stage

The first stage of the problem occurs as online orders arrive at the retailer. The retailer must decide whether to accept or reject each order as it arrives, in an online manner. We are mostly interested in threshold policies, a restriction that reflects real-world policies used by retailers. However, these are not the only policies which could be used, in principle, to solve this first stage decision problem. The first stage concludes after all online orders have arrived and are accepted or rejected by the retailer.

More formally, online orders will arrive during the interval $[0, T]$. At any moment t, the state parameter $\lambda = [\lambda_1, \ldots, \lambda_n]$ contains the set of online orders that have previously been accepted from each location. The first stage problem for the retailer is to set a $0-1$ function f, so that when an order occurs from location i at time t, $f(i, t, \lambda) = 1$ if this order is to be accepted and $f(i, t, \lambda) = 0$ if the order is to be rejected. For most of this work, we restrict f to a class of policies we define as threshold policies. These policies are commonly used in the retail industry and focusing on this policy class makes this problem more approachable.

Threshold Policies

We consider two types of threshold policies, Local Thresholds and Global Thresholds.

Definition 1. *A Local Threshold policy $[S_1, \ldots, S_n]$ accepts the first S_i online orders from location i $\forall i \in [n]$ and rejects all remaining orders.*

Local Threshold policies have a parameter for each store location, allowing the retailer fine-tuned control over which areas are accepting online orders.

Definition 2. *A Global Threshold policy S accepts the first S online orders (from all locations) and rejects all remaining orders.*

Global Threshold policies have a single parameter for the full network of stores. Global Threshold policies allow the retailer more control over the total number of online orders accepted but less fine-tune control than with Local Thresholds over which orders are accepted.

Many retailers already use threshold policies to manage their online sales channels, so it is natural to focus on this policy class. Restricting the retailer to threshold policies also helps us analyze the model. More complex policies might utilize the arrival times of orders, but this is not our focus in this analysis. In Section 1.5 we will explore how a retailer can set these threshold policies.

1.3.3 Second Stage

After the first stage concludes, the retailer learns the amount of in-store demand they received as the online orders arrived. The retailer then must decide whether to cancel or fulfill each accepted online order, and from which store will inventory be used to fill these orders. This problem can be naturally formulated as a network flow problem and is solvable as the following optimization problem:

$$\begin{aligned}
\text{minimize } & p\min\Big(\sum_{i=1}^{n}(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^{n} F_{ij}), \sum_{i=1}^{n}(\mathcal{D}_i^O - A_i^O)\Big) \\
& + \sum_{i=1}^{n}(c_i C_i + \sum_{j=1}^{n} s_{ji} F_{ji}) \\
\text{such that } & \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\
& C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n] \\
& C_i, R_i, F_{ij} \geq 0, \ \forall i, j.
\end{aligned} \tag{1.1}$$

Consider first the objective function:

$$p\min\Big(\sum_{i=1}^{n}(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^{n} F_{ij}), \sum_{i=1}^{n}(\mathcal{D}_i^O - A_i^O)\Big) + \sum_{i=1}^{n}(c_i C_i + \sum_{j=1}^{n} s_{ji} F_{ji}).$$

The expression $\sum_{i=1}^{n}(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^{n} F_{ij})$ is the amount of remaining inventory after all orders have been fulfilled or cancelled. $\mathcal{I}_i$ is the starting inventory at location i, $\mathcal{D}_i^P$ is the in-store demand at location i, and F_{ij} is the number of filled online orders received at location j and filled from inventory at location i. The expression $\sum_{i=1}^{n}(\mathcal{D}_i^O - A_i^O)$ is the number of online orders that were rejected. $\mathcal{D}_i^O$ is the amount of online demand at location i, and A_i^O is the number of online orders accepted from location i in the first stage. Consequently, $\min(\sum_{i=1}^n(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^n F_{ij}), \sum_{i=1}^n(\mathcal{D}_i^O - A_i^O))$ is the number of rejected online orders which could have been fulfilled had they been accepted. The objective function assigns a cost of p to each of these orders, reflecting the sale price of the item. The expression $\sum_{i=1}^n c_i C_i$ reflects the sum of all cancellation penalties orders that are cancelled. c_i is the cost parameter of a cancelled order from location i and C_i is the decision variable for the number of online orders cancelled from location i. Lastly, the expression $\sum_{i=1}^n \sum_{j=1}^n s_{ji} F_{ji}$ represents the shipping costs for all online orders that were accepted and fulfilled. s_{ji} is the shipping cost between locations j and i, and F_{ji} is the decision variable reflecting the number of online orders filled from inventory at location j and shipped to customers in location i.

The constraints of the form $\min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^n F_{ij} = \mathcal{I}_i$ express that all inventory at location i must be used to fulfill in-store demand, be saved, or be used to fulfill online demand. R_i is a decision variable reflecting the amount of inventory that is left over. The constraints of the form $C_i + \sum_{j=1}^n F_{ji} = A_i^O$ reflect that all accepted orders must be either cancelled or fulfilled.

1.3.4 Model Variables and Parameters

The overall optimization problem is to minimize the expected value of second stage problem. The components of this model are the following:

Inputs:

- Online demand $\mathcal{D}_i^O$ for customers at location i, drawn from probability distribution f_i^O
- In-store demand $\mathcal{D}_i^P$ at location i, drawn from probability distribution f_i^P
- Inventory level $\mathcal{I}_i$ at location i
- Cancellation penalty c
- Unnecessary rejection penalty p. The retailer pays a penalty of p for rejecting an order that would have been filled successfully if accepted.
- Shipping costs s_{ij} between locations i and j

Decisions:

- Threshold level S_i at location i (if a threshold policy is in place)
- Filled online orders, F_{ij} received at location j and filled from inventory at location i

Bookkeeping:

- Cancelled online orders C_i for customers at location i

- Accepted online orders A_i^O for customers at location i
- Accepted physical orders A_i^P for customers at location i
- Leftover inventory R_i at location

We first investigate a single-store, known demand distribution setting. Access to the CDFs of the demand distributions allows us to compute a closed form solution that maximizes expected retailer profits. With only a single physical location, the fulfillment problem for accepted online orders is trivial, and we are able to prove optimal reserve thresholds in a similar manner to the the analysis of the classical Newsvendor problem.

Next, we generalize to a multiple-store, unknown demand distribution setting. In addition to determining conditions to open and close the online sales channel, the retailer must now also set a fulfillment policy to ship the accepted online orders to their destinations. Given a fixed threshold policy, we show that the fulfillment problem can be solved as a network flow problem. Furthermore, we can examine certain dual values of a network flow linear program to compute unbiased estimates of the derivative of the reserve thresholds with respect to the retailer's expected profit. This leads to an Infinitesimal Perturbation Analysis (IPA) algorithm [23] we will use to set threshold policies for the retailer. We conclude with numerical experiments that provide insight into when and why these threshold policies are effective for solving this model.

1.4 Single-Store Model

We will consider the special case of our Omni-Channel Fulfillment Model where there is only a single store location and first stage policies are restricted to Local Threshold policies. Note that for a single-location instance Local Threshold and Global Threshold policies are equivalent classes. This restriction simplifies the problem, allowing us to build some intuition for the full model. We will show that single-location restriction means that the second stage problem is straightforward: $\min(\mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P), \min(\mathcal{D}^O, S))$ accepted online orders are filled and the rest are cancelled. In this section, we remove the subscripts denoting store location when discussing single-location instances of the model (S_1 becomes S, etc.). There is no need to solve a linear program to determine a fulfillment assignment. Using this observation, we can re-write the optimization problem in a simpler form. Consider the linear program from the second stage:

$$\begin{aligned}
&\text{minimize } p\min\Big(\sum_{i=1}^{n}(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^{n} F_{ij}), \sum_{i=1}^{n}(\mathcal{D}_i^O - A_i^O)\Big) \\
&\qquad + \sum_{i=1}^{n}(c_i C_i + \sum_{j=1}^{n} s_{ji}F_{ji}) \\
&\text{such that } \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\
&\qquad C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n] \\
&\qquad C_i, R_i, F_{ij} \geq 0, \ \forall i, j.
\end{aligned} \tag{1.2}$$

We will re-write the objective function knowing that we are considering only single store instances (and remove store-location subscripts when appropriate). First, we can remove the summations, replace A_1^O with $\min(\mathcal{D}^O, S)$, and remove shipping costs as we assume $s_{11} = 0$:

$$p\min(\mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P) - F, \mathcal{D}^O - \min(\mathcal{D}^O, S)) + cC.$$

We observe that F will be at most the number of accepted online orders A^O, and F will also be at most the amount of remaining inventory after in-store demand is filled. It will be optimal to maximize F subject to satisfying these constraints as it is preferable to fill all accepted orders for which there is leftover inventory, rather than cancelling these orders.

Proposition 1. *For a single-store instance of the Omni-Channel Fulfillment Model* $F = \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$ *is satisfied in any optimal solution.*

Proof. First, we show that for any feasible solution, (C, R, F),

$$F \leq \min(A^O, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)).$$

By constraint $C + F = A^O$, $F \leq A^O$. Similarly, by constraint $\min(\mathcal{D}^P, \mathcal{I}) + R + F = \mathcal{I}$, $F \leq \mathcal{I} - \min(\mathcal{D}^P, \mathcal{I})$. Then $F \leq \min(A^O, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$.

Now consider an arbitrary feasible solution where this inequality is not tight, $F < \min(A^O, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$. We will demonstrate that such a solution cannot be optimal. In such a solution $F < A^O$ and $F < \mathcal{I} - \min(\mathcal{D}^P, \mathcal{I})$. If $F < A^O$ then $C > 0$, and if $F < \mathcal{I} - \min(\mathcal{D}^P, \mathcal{I})$ then $R > 0$. Then for any $\epsilon < \min(R, C)$, $(C - \epsilon, R - \epsilon, F + \epsilon)$ will be a feasible solution whose objective value is $\epsilon(c+p)$ smaller than the objective value of (C, R, F). Consequently, $F = \min(A^O, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$ in any optimal solution. We recall that $A^O = \min(\mathcal{D}^O, S)$ by the definition of a Local

Threshold policy. Therefore,

$$\begin{aligned} F &= \min(A^O, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)) \\ &= \min(\min(\mathcal{D}^O, S), \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)) \\ &= \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)). \end{aligned}$$

$\square$

By Proposition 1 and the constraint $C + F = A^O$, we see that for any optimal solution, $C = \min(\mathcal{D}^O, S) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$ and $F = \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P))$. Consequently, this re-stated objective function reaches the same value as the original objective function at an optimal solution:

$$\begin{aligned} p \quad & \min(\mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)), \\ & \quad \mathcal{D}^O - \min(\mathcal{D}^O, S) \qquad) \\ +c \quad & \quad (\min(\mathcal{D}^O, S) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)) \qquad). \end{aligned}$$

This function contains none of the decision variables in the constraints of the second stage linear program, but it does depend on the value of threshold parameter S, which is set during the first stage. Consequently, we can write the entire Omni-Channel Fulfillment problem in a single line for the single-store case:

$$\begin{aligned} \min_S E_{\mathcal{D}^O, \mathcal{D}^P}[\quad & p \cdot \min[\mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)), \\ & \quad \mathcal{D}^O - \min(\mathcal{D}^O, S) \qquad] \\ & +c \cdot (\min[\mathcal{D}^O, S] - \min[\mathcal{D}^O, S, \mathcal{I} - \min[\mathcal{I}, \mathcal{D}^P]) \qquad]. \end{aligned}$$

Next, we will prove a closed-form optimal solution to the problem when the cumulative distribution functions (CDFs) of the demand distributions are fully known. In Section 1.8.2 we consider the case where the CDF is not available and we establish a sample complexity bound on the number of samples needed to obtain a high quality solution to the problem with high probability.

I show in Theorem 1 that the optimal threshold for this problem takes a similar form to the solution to the newsvendor problem:

$$S = \mathcal{I} - F_P^{-1}\left(\frac{c}{c+p}\right).$$

One interesting consequence of this theorem is that the optimal threshold depends only on the distribution of physical demand, not online demand.

The proof of the optimal threshold also provides insight on the structure in this optimization problem. The theorem is proved by arguing that $G(S)$, the expected value of the optimization problem as a function of threshold S is unimodal and finding the point where the derivative of $G(S)$ with respect to S changes signs. We compute the derivative of $G(S)$ and observe that this derivative reduces to the expression

$$P[S < \mathcal{D}^O](cP[\mathcal{D}^P \geq \mathcal{I} - S] - pP[\mathcal{D}^P < \mathcal{I} - S])$$

The factor in this expression,

$$cP[\mathcal{D}^P \geq \mathcal{I} - S] - pP[\mathcal{D}^P < \mathcal{I} - S].$$

is nearly identical to the derivative of the classical newsvendor problem's expected objective with respect to the quantity purchased. This reveals a close connection between the omni-channel fulfillment problem studied and the classical newsvendor model. Additionally, it is the presence of the other multiplicative factor, $P[S < \mathcal{D}^O]$, which makes this makes the function $G(S)$ non-convex (though still unimodal).

We observe this empirically by plotting the average objective value over 50000 samples taken at each feasible threshold level. Figures 1.1-1.3 present these results for instances all with $c = 15$, $p = 10$, $I = 30$, $\mathcal{D}^P \sim$ Poisson(20), but $\mathcal{D}^O$ is drawn from Poisson distributions with rate parameters ranging from 10 to 50. It's clear that the optimal threshold value does not depend on the online demand distribution, but this distribution does significantly influence the shape of function $G(S)$ and can result in non-convex $G(S)$ functions when $P[x < \mathcal{D}^O]$ is small for $x < I$. These observations hold in the full, multi-location version of the model, which is why we focus on proving unimodality rather than convexity in Theorem 2.

Theorem 1. *For the above model,*

$$\begin{aligned} \arg\min_S E_{\mathcal{D}^O,\mathcal{D}^P}[\quad & p \cdot \min[\quad \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I}, \mathcal{D}^P)), \\ & \qquad\qquad \mathcal{D}^O - \min(\mathcal{D}^O, S)] \\ & + c\cdot(\qquad \min[\mathcal{D}^O, S] - \min[\mathcal{D}^O, S, \mathcal{I} - \min[\mathcal{I}, \mathcal{D}^P]) \quad] \\ = & \qquad\qquad \mathcal{I} - F_P^{-1}(\frac{c}{c+p}). \end{aligned}$$

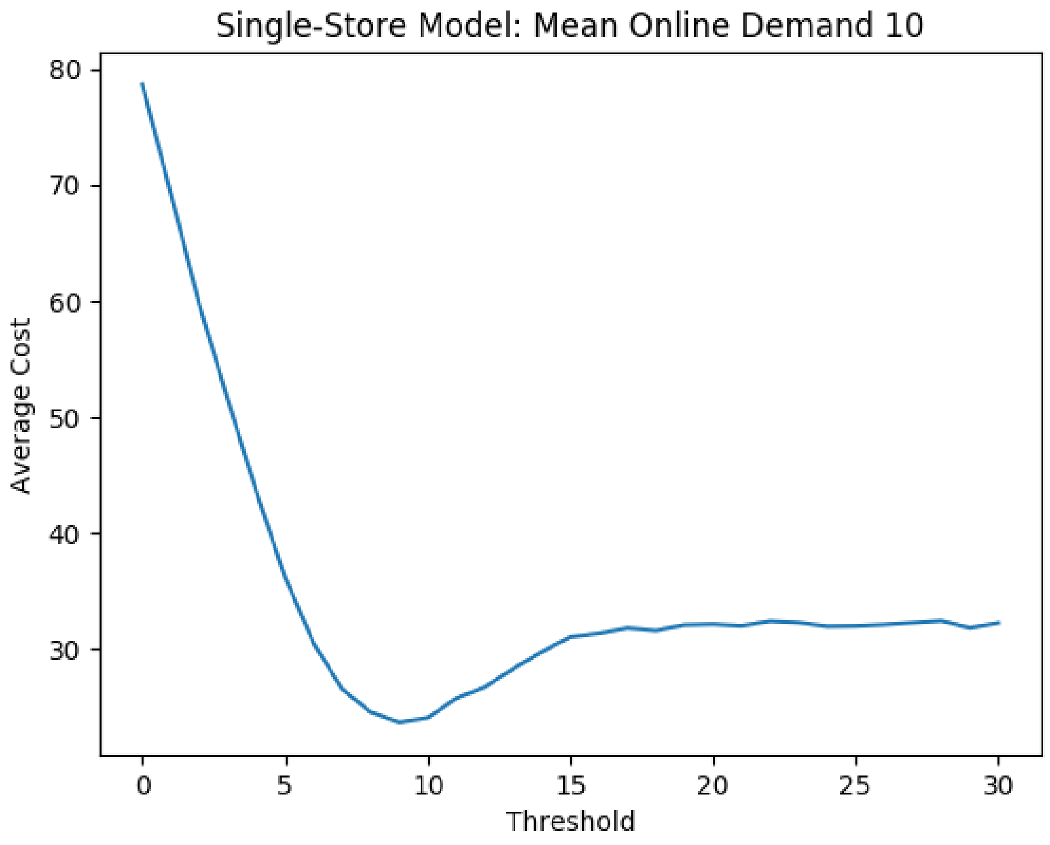


FIGURE 1.1: Plot of $G(S)$ for instance where $\mathcal{D}^O \sim \text{Poisson}(10)$

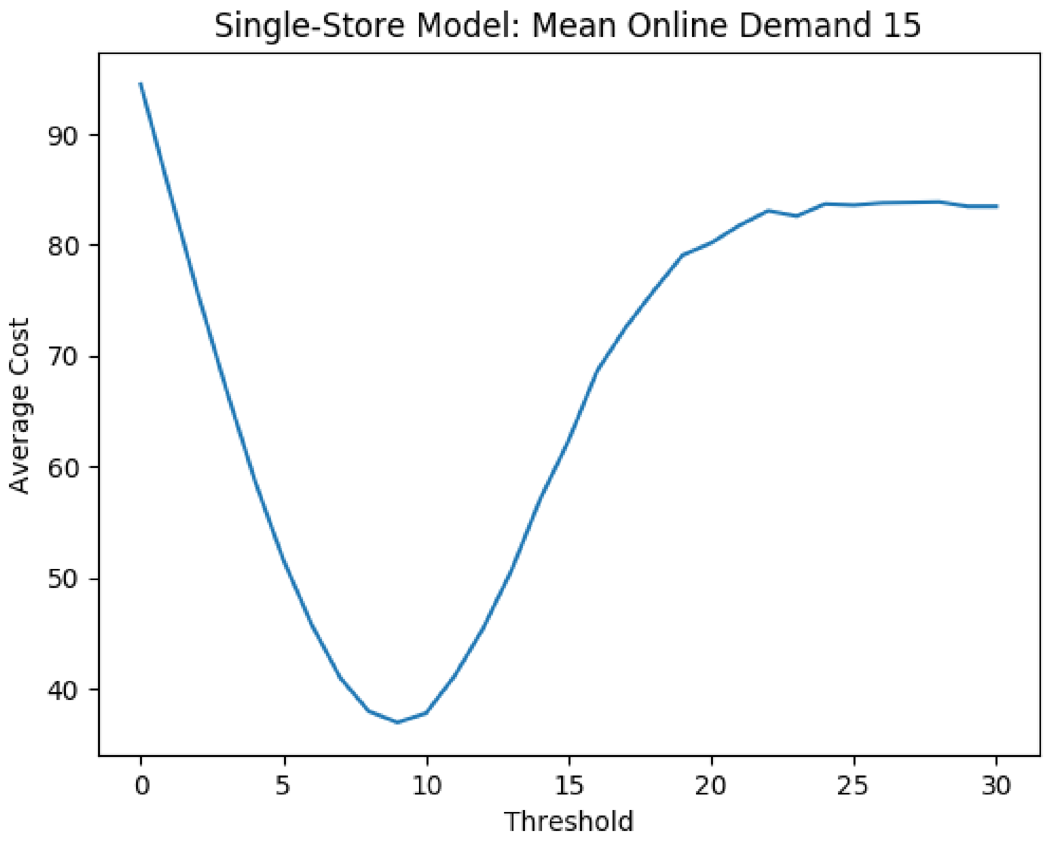


FIGURE 1.2: Plot of $G(S)$ for instance where $\mathcal{D}^O \sim \text{Poisson}(15)$

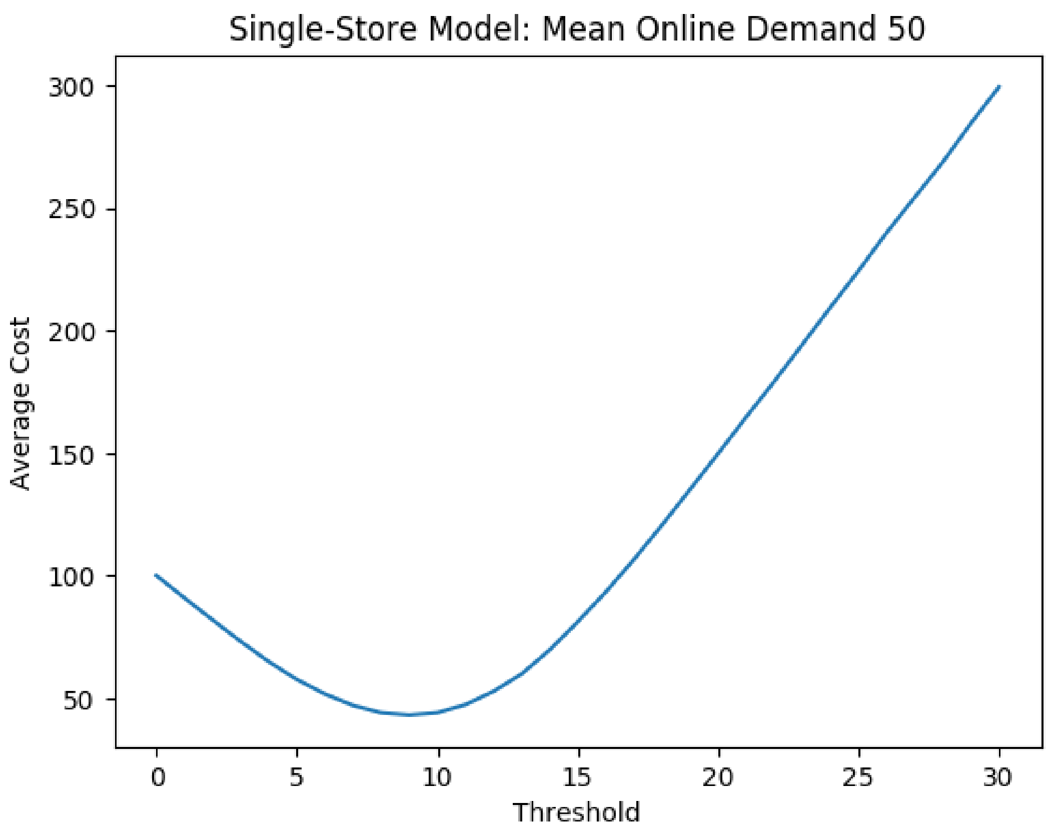


FIGURE 1.3: Plot of $G(S)$ for instance where $\mathcal{D}^O \sim \text{Poisson}(50)$

Proof. Let

$$\begin{aligned} G(S) = & E_{\mathcal{D}^O,\mathcal{D}^P}\Big[\quad p \cdot \min[\mathcal{I} - \min(\mathcal{I},\mathcal{D}^P) - \min(\mathcal{D}^O, S, \mathcal{I} - \min(\mathcal{I},\mathcal{D}^P)), \\ & \qquad \mathcal{D}^O - \min(\mathcal{D}^O, S)] \\ & \qquad + c \cdot (\min[\mathcal{D}^O, S] - \min[\mathcal{D}^O, S, \mathcal{I} - \min[\mathcal{I},\mathcal{D}^P]) \quad \Big]. \end{aligned}$$

We can also observe the derivative of this function with respect to threshold S:

$$\begin{aligned}
&\frac{d}{dS}[E_{\mathcal{D}^O,\mathcal{D}^P}[p\cdot \min[\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)-\min(\mathcal{D}^O,S,\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)),\\
&\qquad\qquad \mathcal{D}^O-\min(\mathcal{D}^O,S)]\\
&\qquad + c\cdot(\min[\mathcal{D}^O,S]-\min[\mathcal{D}^O,S,\mathcal{I}-\min[\mathcal{I},\mathcal{D}^P])]]\\
&=\frac{d}{dS}[\int_0^\infty\int_0^\infty f_P(x)f_O(y)(p\cdot\min[\mathcal{I}-\min(\mathcal{I},x)-\min(y,S,\mathcal{I}-\min(\mathcal{I},x)),\\
&\qquad\qquad y-\min(y,S)]]\\
&\qquad + c\cdot(\min[y,S]-\min[y,S,\mathcal{I}-\min[\mathcal{I},x]])dxdy]\\
&=\int_0^\infty\int_0^\infty f_P(x)f_O(y)(c\cdot\mathbf{1}[\mathcal{I}-\min(\mathcal{I},x)\le S\le y]\\
&\qquad -p\cdot\mathbf{1}[S<\min(y,\mathcal{I}-\min(\mathcal{I},x)])dxdy\\
&=cP[\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)\le S<\mathcal{D}^O]-pP[S<\min(\mathcal{D}^O,\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P))]\\
&=cP[S<\mathcal{D}^O]P[S\ge\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)]-pP[S<\mathcal{D}^O]P[S<\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)]\\
&=P[S<\mathcal{D}^O](cP[S\ge\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)]-pP[S<\mathcal{I}-\min(\mathcal{I},\mathcal{D}^P)])\\
&=P[S<\mathcal{D}^O](cP[\mathcal{D}^P\ge\mathcal{I}-S]-pP[\mathcal{D}^P<\mathcal{I}-S]).
\end{aligned}$$

Then, assuming $0<F_O(x)<1$ for $x\in(0,I)$, $G'(S)=0$ if and only if $pF_P(\mathcal{I}-S)=c(1-F_P(\mathcal{I}-S))$ or equivalently if $\mathcal{I}-S=F_P^{-1}(\frac{c}{c+p})$. Let $S^*=\mathcal{I}-F_p^{-1}(\frac{c}{c+p})$, or equivalently $P[\mathcal{D}^P<\mathcal{I}-S^*]=\frac{c}{c+p}$. If $S<S^*$ then $G'(S)<0$:

$$\begin{aligned}
G'(S)&=P[S<\mathcal{D}^O](cP[\mathcal{D}^P\ge\mathcal{I}-S]-pP[\mathcal{D}^P<\mathcal{I}-S])\\
&=P[S<\mathcal{D}^O](c(1-P[\mathcal{D}^P<\mathcal{I}-S]-pP[\mathcal{D}^P<\mathcal{I}-S])\\
&<P[S<\mathcal{D}^O](c-\frac{c^2}{c+p}-\frac{cp}{c+p})\\
&=0.
\end{aligned}$$

Similarly, if $S>S^*$ then $G'(S)>0$:

$$\begin{aligned}
G'(S)&=P[S<\mathcal{D}^O](cP[\mathcal{D}^P\ge\mathcal{I}-S]-pP[\mathcal{D}^P<\mathcal{I}-S])\\
&=P[S<\mathcal{D}^O](c(1-P[\mathcal{D}^P<\mathcal{I}-S]-pP[\mathcal{D}^P<\mathcal{I}-S])\\
&>P[S<\mathcal{D}^O](c-\frac{c^2}{c+p}-\frac{cp}{c+p})\\
&=0.
\end{aligned}$$

$G()$ is decreasing when $S<S^*$ and increasing when $S>S^*$, so S^* is an optimal threshold.

$\square$

1.5 Infinitesimal Perturbation Analysis Method

We return to the complete omni-channel fulfillment model, now in the multiple-location setting, to present an Infinitesimal Perturbation Analysis (IPA) algorithm that converges to optimal policies for certain policy classes. This IPA method can be used to obtain Global Threshold and Local Threshold policies for the Omni-Channel Fulfillment Model. Recall that this problem has two stages: the first stage problem is to accept or reject incoming online orders, and the second stage problem is to assign accepted online orders to stores for fulfillment and cancel any unfulfilled orders. The key insight behind our IPA method is that we can use dual values of a linear program related to the second stage problem to produce unbiased estimates of the derivative of the objective function with respect to threshold parameters. We apply this method to set both Global and Local Threshold policies.

1.5.1 Overview

We focus on two policy classes for the first stage, Local Thresholds and Global Thresholds. Recall that a Local Threshold policy has a threshold value, S_i, for each store location, and the first S_i online orders are accepted for each location i and all remaining orders are rejected. A Global Threshold policy has a single threshold value S so that the first S orders (from any location) are accepted and all remaining orders are rejected. We will use essentially the same IPA algorithm to find optimal local and global threshold policies.

After all online orders have been accepted or rejected, the accepted orders must be assigned to store locations for fulfillment or cancelled. This assignment problem can be formulated as a linear program. We will use dual values from this linear program in our IPA method to optimize both Global and Local Thresholds, so we begin by formulating and describing the second stage problem. Then, we will develop the optimization procedure for each threshold policy class.

1.5.2 Second Stage Assignment Problem

Original minimization assignment problem

Recall the original assignment problem defined in Section 1.3.3:

$$
\begin{aligned}
&\min p \min\Big(\sum_{i=1}^{n}(\mathcal{I}_i - \min(\mathcal{I}_i, \mathcal{D}_i^P) - \sum_{j=1}^{n} F_{ij}), \sum_{i=1}^{n}(\mathcal{D}_i^O - A_i^O)\Big) \\
&\quad + \sum_{i=1}^{n}(c_i C_i + \sum_{j=1}^{n} s_{ji} F_{ji}) \\
&\text{such that} \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\
&\quad C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n] \\
&\quad C_i, R_i, F_{ij} \geq 0, \ \forall i, j.
\end{aligned}
\tag{1.3}
$$

The objective function of this LP reflects the number of rejected orders which could have been filled with leftover inventory, the number of cancelled orders at each location, and the shipping costs associated with filling online orders. The first set of constraints requires all inventory to be sold in-store, salvaged, or used to fill online demand. The second set of constraints requires all accepted online orders to be cancelled or filled. We intend to compute derivative estimates using the dual values of this constraint set. However, we cannot do this with the above LP because the value that changes when relaxing the right-hand side of this constraint, A_i^O, also appears in the LP's objective function.

We formulate an alternative LP with equivalent optimal solutions. This will allow us to compute gradient estimates in a more straightforward manner. Through this transformation we will obtain the following linear program:

$$
\begin{aligned}
&\max \sum_{i=1}^{n}(p \min(\mathcal{D}_i^P, \mathcal{I}_i) - c_i C_i + \sum_{j=1}^{n}(p - s_{ji}) F_{ji}) \\
&\text{such that} \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\
&\quad C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n] \\
&\quad C_i, R_i, F_{ij} \geq 0, \ \forall i, j.
\end{aligned}
\tag{1.4}
$$

We can immediately observe it has an economic interpretation consistent with the original LP 1.3, and we will go on to show that this LP is equivalent for the purpose of computing derivative estimates. The first term in the objective function, $\sum_{i=1}^{n}(p \min(\mathcal{D}_i^P, \mathcal{I}_i)$, reflects a profit of p for each unit sold in-store. The remaining terms of the objective function, $-c_i C_i + \sum_{j=1}^{n}(p - s_{ji}) F_{ji}$, reflect a profit of p for each unit sold online, with costs deducted for cancellations and shipping costs. This interpretation of LP 1.4 may at first seem counterintuitive because p was originally defined as the penalty cost for missed sales. However, we can observe that the damage incurred to the retailer from missing a sale is the profit they would get from making an additional sale, which explains why the economic interpretations of LPs 1.3 and 1.4 are consistent with each other. Both interpretations implicitly assume that the retailer gets no value from holding on to excess inventory. In the event that this is an unrealistic assumption, it is very straightforward to incorporate a salvage value of inventory into the model.

Proposition 2. *For any optimal solution to the original minimization LP 1.3 there is an optimal solution to the maximization LP 1.4 that yields an identical assignment of orders to stores.*

Proof. Consider again the objective function in the original minimization LP 1.3:
$p\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij}),\sum_{i=1}^n(\mathcal{D}_i^O-A_i^O))+\sum_{i=1}^n(c_iC_i+\sum_{j=1}^n s_{ji}F_{ji})$
We'll focus on the first term,

$$\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij}),\sum_{i=1}^n(\mathcal{D}_i^O-A_i^O)),$$

which reflects the amount of leftover inventory that could have been used to meet unfilled demand. By the constraints of the LP, $C_i+\sum_{j=1}^n F_{ji}=A_i^O$, so we can rewrite this as $\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij}),\sum_{i=1}^n(\mathcal{D}_i^O-C_i-\sum_{j=1}^n F_{ji}))$. We assume in the original formulation that if any cancellations occur, then there is no remaining inventory. This is equivalent to assuming that $p-\max_{i,j\in[n]} s_{ij}>c$. In other words, the maximum ship cost in the network is small enough that it is always preferable to fill an online order rather than cancel the order. Note that once we have reformulated the problem we will be able to drop this assumption. Consequently, if $C_i>0$ then $\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij})=0$. This is the crucial observation that leads to the precise correspondence between LPs 1.3 and 1.4. $C_i+\sum_{j=1}^n F_{ji}=A_i^O\le\mathcal{D}_i^O$ so $\sum_{i=1}^n(\mathcal{D}_i^O-C_i-\sum_{j=1}^n F_{ji})\ge 0$. Therefore, $\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij}),\sum_{i=1}^n(\mathcal{D}_i^O-C_i-\sum_{j=1}^n F_{ji}))$ is equivalent to $\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)-\sum_{j=1}^n F_{ij}),\sum_{i=1}^n(\mathcal{D}_i^O-\sum_{j=1}^n F_{ji}))$ for all feasible solutions to the LP. We can rewrite this as $\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)),\sum_{i=1}^n\mathcal{D}_i^O)-\sum_{i=1}^n\sum_{j=1}^n F_{ji})$.

We rewrite the objective function using this reformulation and get the following:

$$p\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)),\sum_{i=1}^n\mathcal{D}_i^O)+\sum_{i=1}^n(c_iC_i+\sum_{j=1}^n(s_{ji}-p)F_{ji}).$$

Observe that the first term, $p\min(\sum_{i=1}^n(\mathcal{I}_i-\min(\mathcal{I}_i,\mathcal{D}_i^P)),\sum_{i=1}^n\mathcal{D}_i^O)$ no longer contains any decision variables, and all decision variables are part of the remaining terms $\sum_{i=1}^n(c_iC_i+\sum_{j=1}^n(s_{ji}-p)F_{ji})$. The first term is important for the economic interpretation of this LP, but only the later terms impact the quality of a feasible solution. Then, maximizing the negative of this function and adding a constant will result in an equivalent problem. This is results in the maximization LP 1.4:

$$\begin{aligned} &\max \sum_{i=1}^{n} (p \min(\mathcal{D}_i^P, \mathcal{I}_i) - c_i C_i + \sum_{j=1}^{n} (p - s_{ji}) F_{ji}) \\ &\text{such that} \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\ &C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n] \\ &C_i, R_i, F_{ij} \geq 0, \ \forall i, j. \end{aligned} \tag{1.5}$$

□

This correspondence means that an adjustment to the constraints of the maximization problem leads to the same improvement in the objective function of the minimization problem as the maximization LP. Then, we can use the dual values the Maximization LP to optimize our policy in the First Stage problem.

The original minimization problem placed assumptions that the cancel cost must be high enough that the retailer would never want to cancel an order they could possibly fill. Without this assumption, it is unclear what it means to "unnecessarily reject" an order, because there could be orders which could theoretically be filled but this would not be the profit-maximizing decision for the retailer. This new LP formulation models the full profits received by the retailer from both online and in-store sales, rather than just costs, so it is no longer necessary to make this assumption if we wish to only use this formulation of the second-stage problem. There are advantages still to the original problem, including that a cost-based objective function can make percentage changes in the objective function more interpretable. For example, if we add k to each I_i as well as shift the probability distributions of $\mathcal{D}_i^P$ up by k, this will increase all realizations of the maximization assignment problem's objective function by pk, whereas the original minimization problem's objective function would not change. Fortunately, if we wish we can still use the original problem as our second stage problem while using this reformulation to get information for our IPA method.

Our interest in the maximization LP 1.4 is so we can extract sensitivity information from the constraints $C_i + \sum_{j=1}^{n} F_{ji} = A_i^O, \ \forall i \in [n]$ using LP dual values. To access this sensitivity information, we will express these constraints as inequality constraints:

$$C_i + \sum_{j=1}^{n} F_{ji} \leq A_i^O, \ \forall i \in [n]$$
$$C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O, \ \forall i \in [n].$$

Then, we will dualize the first set of inequalities to obtain the following LP:

$$\begin{aligned}
&\max \sum_{i=1}^{n}(p\min(\mathcal{D}_i^P,\mathcal{I}_i) - (c_i+M)C_i + MA_i + \sum_{j=1}^{n}(p - s_{ji} - M)F_{ji}) \\
&\text{such that} \min(\mathcal{D}_i^P,\mathcal{I}_i) + R_i + \sum_{j=1}^{n} F_{ij} = \mathcal{I}_i,\ \forall i \in [n] \\
&\qquad C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O,\ \forall i \in [n] \\
&\qquad C_i, R_i, F_{ij} \geq 0,\ \forall i,j.
\end{aligned} \tag{1.6}$$

We show in Proposition 3 that this LP 1.6 has the same optimal solution as the original second-stage and maximization LPs. This means we will be able to use the dual values of constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O,\ \forall i \in [n]$ to obtain gradient estimates for the Omni-Channel Fulfillment Model.

Proposition 3. *The linear program 1.6 has the same optimal solution as the linear program 1.4 when $M > p$.*

Proof. By moving the constraints $C_i + \sum_{j=1}^{n} F_{ji} \leq A_i^O,\ \forall i \in [n]$ to the objective function, we are relaxing the linear program 1.4 by allowing solutions that pay a penalty of M for each unit of violation of constraints $C_i + \sum_{j=1}^{n} F_{ji} \leq A_i^O,\ \forall i \in [n]$. The cancel variables C_i only appear in the the order acceptance constraints and have a negative coefficient in the objective, even with the violation penalty M removed. Consequently, any solution where any of constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O,\ \forall i \in [n]$ have slack and any variable $C_i > 0$ is not an optimal solution because it can be improved by decreasing C_i by a sufficiently small value ϵ.

If constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O,\ \forall i \in [n]$ have slack, $C_i = 0\ \forall i \in [n]$. In this case, there necessarily exists i, j such that $F_{ij} > 0$, and we will show that this solution also cannot be optimal. The solution obtained by reducing F_{ij} by ϵ and increasing R_i by ϵ (for a value of ϵ smaller than the slack in the $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O$ constraint) will be a feasible solution with an objective value at least $M - p > 0$ greater than the prior solution. Consequently, no optimal solution will have slack in any of constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O,\ \forall i \in [n]$ and therefore this solution will also be feasible in linear program 1.4. □

Proposition 4. *The minimization LP 1.3 is integral.*

Proof. The integrality of the maximization LP 1.4 follows from the observation that this LP models a minimum cost feasible flow problem. The network shown in Figure 1.4 has exact flow requirements indicated by the edge labels and unrestricted capacity on all unlabeled edges. Flow out of the top node on the left column of nodes represents cancelations, and all flow exiting this node incurs a cost of c. Flow into the top node on the right column of nodes represents salvaged inventory. There is no cost to send flow to this node and has unlimited capacity. The remaining nodes in the left column represent remaining inventory at each store after in-store demand is filled, and the remaining nodes in the right column represent accepted online orders. Flow from a left inventory node $I_{i,a}$ to a right inventory node $I_{j,b}$ represents inventory at location i used to fill orders at location j and has a cost $s_{i,j} - p$.

We will demonstrate that this minimum cost flow problem is equivalent to minimization LP 1.3 by writing out the flow problem as a linear program. Before we write the complete flow LP, let's enumerate its constraints moving from left to right Figure 1.4. The first layer of edges sets and exact flow constraint from node s to each node $I_{i,a}$ for all locations i. These constraints can be expressed as

$$x_{s,I_{i,a}} = \mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i), \ \forall i \in [n]. \tag{1.7}$$

The next layer of nodes in the network has no capacity constraints, but each node has flow conservation constraints. We consider first the flow conservation constraint on node C:

$$x_{s,C} = \sum_{i=1}^{n} x_{C,I_{i,b}}. \tag{1.8}$$

The flow conservation constraints on the left inventory nodes are

$$x_{s,I_{i,a}} = x_{I_{i,a},R} + \sum_{j=1}^{n} x_{I_{i,a},I_{j,b}}, \ \forall i \in [n]. \tag{1.9}$$

There are no edge capacities on the central edges, so we move on to the flow capacity constraints on the right side of nodes. The flow conservation constraint on node R is

$$\sum_{i=1}^{n} x_{I_{i,a},R} = x_{R,t}. \tag{1.10}$$

The flow conservation constraints on the right inventory nodes are

$$x_{C,I_{i,b}} + \sum_{j=1}^{n} x_{I_{j,a},I_{i,b}} = x_{I_{i,b},t}, \ \forall i \in [n]. \tag{1.11}$$

The edge capacity constraints from the right inventory nodes to t is

$$x_{I_{i,b},t} = A_i, \ \forall i \in [n]. \tag{1.12}$$

Finally, flow conservation constraint on nodes s and t are

$$x_{t,s} = x_{s,C} + \sum_{j=1}^{n} x_{s,I_{j,a}} \tag{1.13}$$

$$x_{R,t} + \sum_{j=1}^{n} x_{I_{j,b},t} = x_{t,s}. \tag{1.14}$$

$x_{t,s}$ is otherwise unconstrained, so we replace this set of constraints with the following set of constraint:

$$x_{s,C} + \sum_{j=1}^{n} x_{s,I_{j,a}} = x_{R,t} + \sum_{j=1}^{n} x_{I_{j,b},t}. \tag{1.15}$$

Now we will reduce these constraints to the set of constraints in the minimization LP 1.3. Variable $x_{s,I_{i,a}}$ is set to a fixed value by constraint 1.7 so we will replace $x_{s,I_{i,a}}$ with $\mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i)$ in all other constraints. We will rename variables $x_{I_{i,a},I_{j,b}}$ to $F_{i,j}$, variable $x_{C,I_{i,b}}$ to C_i, and $x_{I_{i,a},R}$ to R_i, $\forall i \in [n]$. We now re-state the constraints, using new variable names and replacing all variables that are constrained to a fixed value with that value:

$$x_{s,C} = \sum_{i=1}^{n} C_i \tag{1.16}$$

$$\mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i) = R_i + \sum_{j=1}^{n} F_{i,j}, \ \forall i \in [n] \tag{1.17}$$

$$\sum_{i=1}^{n} R_i = x_{R,t} \tag{1.18}$$

$$C_i + \sum_{j=1}^{n} F_{j,i} = A_i, \ \forall i \in [n] \tag{1.19}$$

$$x_{s,C} + \sum_{j=1}^{n} \mathcal{I}_j - \min(\mathcal{D}_j^P, \mathcal{I}_j) = x_{R,t} + \sum_{j=1}^{n} A_j. \tag{1.20}$$

We can consolidate constraints 1.16, 1.18, and 1.20 into a single constraint:

$$\sum_{i=1}^{n} C_i + \mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i) = \sum_{i=1}^{n} R_i + A_i. \tag{1.21}$$

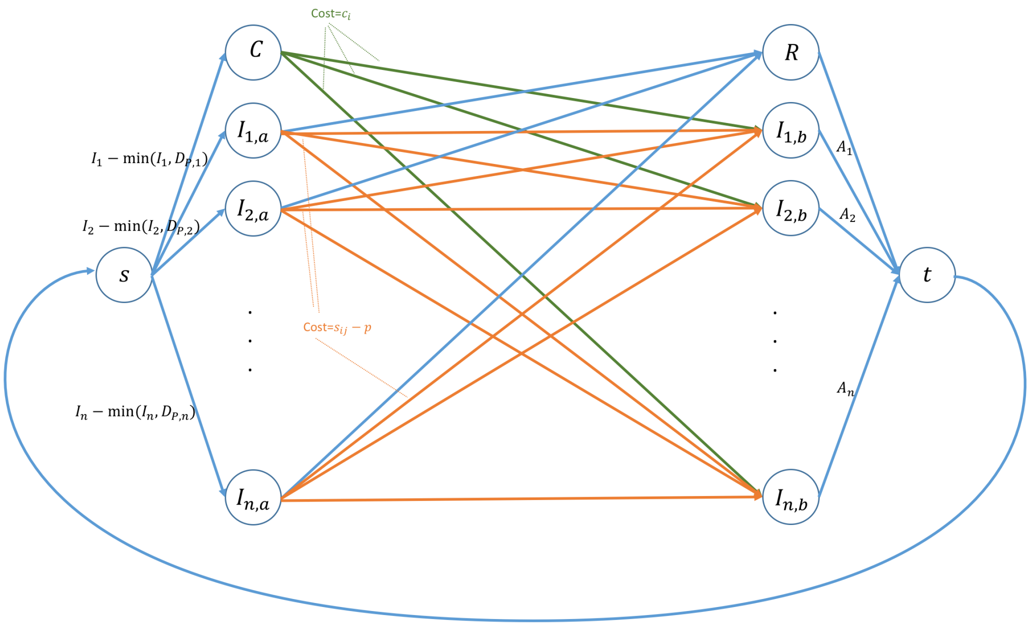


FIGURE 1.4: Minimum cost flow formulation of maximization LP 1.4

This results in the following constraint set for the minimum cost flow problem shown in Figure 1.4:

$$\mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i) = R_i + \sum_{j=1}^{n} F_{i,j}, \ \forall i \in [n] \tag{1.22}$$

$$C_i + \sum_{j=1}^{n} F_{j,i} = A_i, \ \forall i \in [n] \tag{1.23}$$

$$\sum_{i=1}^{n} \mathcal{I}_i - \min(\mathcal{D}_i^P, \mathcal{I}_i) = \sum_{i=1}^{n} R_i + A_i - C_i. \tag{1.24}$$

Constraints 1.22 and 1.23 are exactly the constraints of minimization LP 1.3. Constraint 1.24 is redundant and is implied by constraints 1.22 and 1.23. Finally, we must verify that the objective functions of these two problems are the same or shifted by a constant. We will write out the objective function of the minimum cost flow problem:

$$\sum_{i=1}^{n} c_i C_i + \sum_{j=1}^{n} (s_{ij} - p) F_{ij}.$$

We observed in the proof of Proposition 2 that this is equal to a constant plus the objective function of minimization LP 1.3. Therefore, the minimization LP 1.3 describes the minimum cost flow problem shown in Figure 1.4 and consequently is integral.

□

The integrality of this LP is important because fractional solutions do not correspond to acceptable real-world fulfillment plans in our discrete model.

1.5.3 First Stage Decision Problem

Online orders arrive in sequence and must be accepted or rejected at the time of arrival. We assume that in-store and online demand are drawn from fixed distributions and the sequence in which online orders arrive is drawn uniformly at random from all orderings of the given demand realization. We consider two policy classes for this decision problem, Local Thresholds and Global Thresholds. The IPA algorithm to optimize these policy classes is similar and relies on the same techniques.

IPA Algorithm Overview

This IPA algorithm can be accurately interpreted as a stochastic gradient descent method. We begin by specifying a starting policy P, and a value U, which will be the number of samples we use to compute a single gradient estimation iteration. U demand samples are drawn, and online orders are accepted and rejected according to policy P for each of the U samples. We assume that policy P is a local or global threshold policy, but the method may apply to additional policy classes. Then, we solve the maximization assignment LP to fulfill the accepted orders in each of the demand samples. Dual values of the maximization assignment LP are used to compute unbiased estimates of the gradients of the fulfillment profit with respect to the policy parameters (S_i in the case of local thresholds). We then update the threshold parameters with their gradients, multiplied by a step size value.

Local Threshold Derivative Estimates

We compute derivative estimates by looking at the dual values corresponding to the LP constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O, \ \forall i \in [n]$ from linear program 1.6, where A_i^O and C_i are the number of accepted and cancelled online orders at location i, respectively, and F_{ji} is the number of online orders at location i filled from inventory from store j. The dual value from one of these constraints indicates the rate of increase in the objective function from relaxing the constraint. For the case of local threshold policies, if demand at location i exceeds threshold S_i then this dual value is precisely the gradient on the total profit of the LP with respect to threshold S_i. We average these gradient estimates over the U samples to get an unbiased estimate of the gradient each time we update the threshold values.

Global Threshold Derivative Estimates

We use the same dual values used to estimate derivatives with respect to Local Threshold parameters, those corresponding to constraints $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O, \ \forall i \in$

$[n]$ from linear program 1.6, to estimate derivatives of the objective function with respect to a Global Threshold parameter. The dual value from one of these constraints indicates the rate of increase in the objective function from relaxing the constraint. For a global threshold policy, if total demand exceeds threshold S, then the derivative of the total profit of the LP with respect to threshold S is the sum of these dual values, weighted by the probabilities that first rejected order is from each store.

Lemma 1. $\frac{dG(S)}{dS} = E[\sum_{i=1}^{n} \frac{\lambda_i}{\sum_{j=1}^{n}\lambda_j} g(i)\mathbf{1}[S < \sum_{k=1}^{n} \mathcal{D}_i^O]]$ *where* λ_i *is the mean online demand at location* i*,* f_i^O*, the distribution from which random variable* $\mathcal{D}_i^O$ *is drawn, is a Poisson distribution, and* $g(i)$ *is the dual value of constraint* $C_i + \sum_{j=1}^{n} F_{ji} \geq A_i^O$ *after solving the maximization LP 1.6 for a demand sample, and* $G(S)$ *is the expected value of the Omni-Channel Fulfillment Model with Global Threshold* S*.*

Proof. Among instances when the first rejected order is at location i, $g(i)$ is the unbiased derivative estimate, so in general, the unbiased derivative estimate is the weighted sum of dual values across all locations, weighted by arrival probability. □

We average these gradient estimates over the U samples to get an unbiased estimate of the gradient each time we update the threshold values.

1.5.4 Threshold Policy Properties

In this section we prove structural properties about threshold policies for the Omni-Channel Fulfillment Model. Theorem 2 states that the expected value of the Model as a function of a Global Threshold is unimodal. A consequence of Theorem 2 is that our IPA method will converge to the optimal policy within this class. We use the same proof technique to show that when all but one S_i of a Local Threshold Policy is fixed, the expected value of the Model as a function of the free Local Threshold parameter is unimodal.

The intuition behind these proofs comes from extending our observations about single-store instances in Section 1.4 to the more general multiple-location setting. We observed that the optimal threshold policy for single-store instances is the $\frac{c}{c+p}$-fractile of the in-store demand distribution. The online demand distribution influences the shape of the expected cost as a function of the threshold, though it does not influence the value of the optimal threshold. An informative way to think about this property is to consider the marginal effect on cost with respect to the threshold. For realizations where online demand is below the threshold this marginal effect is zero, and this marginal effect will have the same non-zero value for all realizations where online demand is above the threshold. Then, the sign of this marginal effect on cost is determined entirely by the demand distribution restricted to realizations where online demand is greater than the threshold value.

We lift these observations to the multiple-store setting and use them to analyze the marginal effect of increasing the threshold. This marginal effect on penalties for missed sale opportunities is negative and decreasing in magnitude, and we use Lemma 2 to establish that the marginal effect on fulfillment costs (including cancel costs) is also always increasing with respect to the threshold. These properties are sufficient to argue that the total expected cost as a function of Global Threshold S (or Local Threshold S_k at location k with all other Local Threshold elements fixed) is unimodal.

Lemma 2. *In a minimum-cost single-commodity flow problem with multiple sources, one sink, and integer supplies, demands, and capacities, the objective value of a minimum cost feasible flow as a function of the supplies at the source nodes is supermodular.*

Proof. Let $s = (s_1, \dots, s_n)$ be the vector of supply at the source nodes, and let $V(s) = V(s_1, \dots, s_n)$ be the objective value of the minimum cost flow in a fixed network as a function of s. We will complete the proof by arguing that

$$\begin{aligned}
&V(s_1+1, s_2+1, \dots, s_n) - V(s_1, s_2+1, \dots, s_n) \\
\geq &V(s_1+1, s_2, \dots, s_n) - V(s_1, s_2, \dots, s_n).
\end{aligned}$$

In other words, increasing the supply by one unit at a supply node cannot decrease the marginal cost of increasing the supply at another node. First observe that the marginal cost of increasing the supply at a specific supply node is the cost of the shortest path in the residual network obtained by computing the minimum cost flow in the network without this additional unit of demand. Suppose that the above inequality is not always true and there is a network where

$$\begin{aligned}
&V(s_1+1, s_2+1, \dots, s_n) - V(s_1, s_2+1, \dots, s_n) \\
< &V(s_1+1, s_2, \dots, s_n) - V(s_1, s_2, \dots, s_n).
\end{aligned}$$

Then adding a unit of supply to source node 2 decreases the marginal cost of adding supply to source node 1. For this to happen, the shortest path in the residual network between source node 1 and the sink under supplies $(s_1, s_2, \dots, s_n)$ must be different from the shortest path in the residual network between source node 1 and the sink under supplies $(s_1, s_2+1, \dots, s_n)$. This requires the shortest path from source node 2 and the sink in the residual network under supplies $(s_1, s_2, \dots, s_n)$ to create a new arc in the residual network by reversing flow in a saturated arc. Then, the shortest path from source node 1 and the sink in the residual network under supplies $(s_1, s_2+1, \dots, s_n)$ must use this newly created arc. This cannot happen, however, because if this newly created path in the residual network (after adding supply to source node 2) is cheaper than the original path from source node 1 to the sink, this contradicts the fact that the augmenting path taken from source node 2 to the sink is a minimum cost path in that residual network. □

Theorem 2. *$G(S)$, the expected value of the objective function of the Omni-Channel Fulfillment Model as a function of Global Threshold S, is unimodal.*

Proof. Let S^* be an optimal solution to the Omni-Channel Fulfillment Model, a solution that minimizes $G()$. We want to show that $G(S+1) - G(S) \leq 0\, \forall S < S^*$ and $G(S+1) - G(S) \geq 0\, \forall S > S^*$. Our proof strategy will be to decompose $G(S)$ into the sum of two functions $P(S)$ and $F(S)$. Then, we will define functions $G(S, D)$, $P(S, D)$, and $F(S, D)$, which are the functions $G(S)$, $P(S)$, and $F(S)$ under arbitrary demand distributions $D = (\mathcal{D}^O, \mathcal{D}^P)$, which may differ from the true demand distributions. $P(S, D)$ represents the "missed sales" cost component of the model and $F(S, D)$ represents the fulfillment costs component of the model.

First, we observe that by the assumptions of our model, in the optimal solution accepted orders will be cancelled only if there is no available inventory to fulfill the orders. Consequently, $\sum_{i,j} F_{ij} = \min(S, \sum_{i=1}^n \mathcal{D}_i^O, \sum_{i=1}^n (I_i - \min(I_i, \mathcal{D}_i^P)))$. Then $P(S, D) = E_D[p \min(\sum_{i=1}^n (I_i - \min(I_i - \mathcal{D}_i^P)) - \min(S, \sum_{i=1}^n \mathcal{D}_i^O, \sum_{i=1}^n (I_i - \min(I_i, \mathcal{D}_i^P))), \sum_{i=1}^n (\mathcal{D}_i^O - A_i^O)]$. Similarly, we can remove the missed sales term from the objective function of the second stage problem and the optimal solution will not change:

$$\begin{aligned}
&\min \sum_{i=1}^n (c_i C_i + \sum_{j=1}^n s_{ji} F_{ji}) \\
&\text{such that} \min(\mathcal{D}_i^P, \mathcal{I}_i) + R_i + \sum_{j=1}^n F_{ij} = \mathcal{I}_i, \ \forall i \in [n] \\
&C_i + \sum_{j=1}^n F_{ji} = A_i^O, \ \forall i \in [n] \\
&C_i, R_i, F_{ij} \geq 0, \ \forall i, j.
\end{aligned} \tag{1.25}$$

Now, let $D(S)$ be the true demand distribution restricted to only outcomes where $\sum_{i=1}^n \mathcal{D}_i^O > S$. We observe that $G(S+1) > G(S)$ if and only if $G(S+1, D(S)) > G(S, D(S))$ and likewise $G(S+1) < G(S)$ if and only if $G(S+1, D(S)) < G(S, D(S))$. This is true because for specific realizations of demand where $\sum_{i=1}^n \mathcal{D}_i^O \leq S$ the value of the model will be equal for Global Thresholds S and $S+1$. Then the realizations of demand where $\sum_{i=1}^n \mathcal{D}_i^O > S$ are the only ones needed to determine whether $G(S+1)$ is greater or smaller than $G(S)$.

To complete the proof, we will show first show that $P(S+1, D(S)) - P(S, D(S))$ and $F(S+1, D(S)) - F(S, D(S))$ are increasing with S. Consequently, $G(S+1, D(S)) - G(S, D(S))$ is also increasing with S. We will use this to show that $G(S+1) - G(S) \leq 0\, \forall S < S^*$ and $G(S+1) - G(S) \geq 0\, \forall S \geq S^*$, concluding the proof.

Observe that $P(S+1, D(S)) - P(S, D(S)) = -p \cdot Pr(\sum_{i=1}^n \min(I_i, \mathcal{D}_i^P) + S < \sum_{i=1}^n I_i)$. This expression is clearly increasing with S and so $P(S+1, D(S)) - P(S, D(S))$ is increasing with S. Next we show that $F(S+1, D(S)) - F(S, D(S))$ is also increasing with S. Linear Program 1.25 may be viewed as a minimum cost single-commodity flow problem in a bipartite network where nodes corresponding to each location are on one side of the bipartition and have supplies equal to the number of accepted orders at that location. Nodes corresponding to each location and a node corresponding to cancellations are on the other side of the bipartition. This second set of nodes all have arcs directed to a sink node, and these arcs have capacities equal to the number of unsold units of inventory at the corresponding location and an unlimited capacity on the arc between the cancellation node and the sink. The sink node has demand equal to the sum of the supplies on the nodes representing accepted orders. This network flow problem is displayed in Figure 1.5.

$F(S+1, D(S)) - F(S, D(S))$ is the difference in expected value between fulfillment costs from accepting $S+1$ and S orders restricted to demand instances where there are at least $S+1$ online orders. An immediate consequence of Lemma 2 is that the marginal cost of fulfilling an order o in addition to a set of orders O, the difference between the minimum possible fulfillment cost of some set of orders O and the minimum possible fulfillment cost of orders $O + \{o\}$, is at least as large as the marginal cost of fulfilling order o in addition to any subset of O. Suppose we have sets M and N of orders whose locations are selected at random from a common probability distribution and $|N| > |M|$. Then the expected marginal cost of filling an order o in addition to orders N will be greater than the expected marginal cost of filling order o in addition to set M. This is a consequence of the supermodularity property seen in Lemma 2 and because the probability distribution of the first $|M|$ order locations in set N is the same as the distribution of order locations in set $|M|$. That $F(S+1, D(S)) - F(S, D(S))$ is increasing with S follows from the previous observation if we consider the case when $|N| = |M| + 1$ and apply the observation over the probability distribution of possible orders o.

We have seen that both $P(S+1, D(S)) - P(S, D(S))$ and $F(S+1, D(S)) - F(S, D(S))$ are increasing with S and so $G(S+1, D(S)) - G(S, D(S))$ is also increasing with S. By assumption that S^* is the optimal Global Threshold, $G(S^*+1) - G(S^*) \geq 0$. Then $G(S^*+1, D(S^*)) - G(S^*, D(S^*)) \geq 0$ and consequently $G(S+1, D(S)) - G(S, D(S)) \geq 0$ and likewise $G(S+1) - G(S) \geq 0\ \forall S \geq S^*$. Similarly, $G(S^*) - G(S^*-1) \leq 0$. Then $G(S^*, D(S^*-1)) - G(S^*-1, D(S^*-1)) \leq 0$ and consequently $G(S, D(S-1)) - G(S-1, D(S-1)) \geq 0$ and likewise $G(S) - G(S-1) \leq 0$ $\forall S \leq S^*$. This concludes the proof as we have proved that $G(S)$ is decreasing at all values of S below S^* and that $G(S)$ is increasing at all values of S above S^*. □

We can apply a similar argument to show that when all but one S_i of a Local Threshold Policy is fixed, the expected value of the Model as a function of the free Local Threshold parameter is unimodal.

Theorem 3. *$G(S_k)$, the expected value of the objective function of the Omni-Channel Fulfillment Model as a function of Local Threshold S_k, is unimodal, when all other Local Threshold parameters, S_j for $j \neq k$ are fixed values.*

Proof. The proof of this theorem follows from nearly the identical argument as was used to prove Theorem 2. Function $G(S_k)$ is decomposed into the sum of $P(S_k)$ and $F(S_k)$. Note that the values of $G(S_k)$, $P(S_k)$, and $F(S_k)$ depend on all Local Thresholds S_i, $\forall i \in [n]$, but we express these functions as functions of S_k to indicate

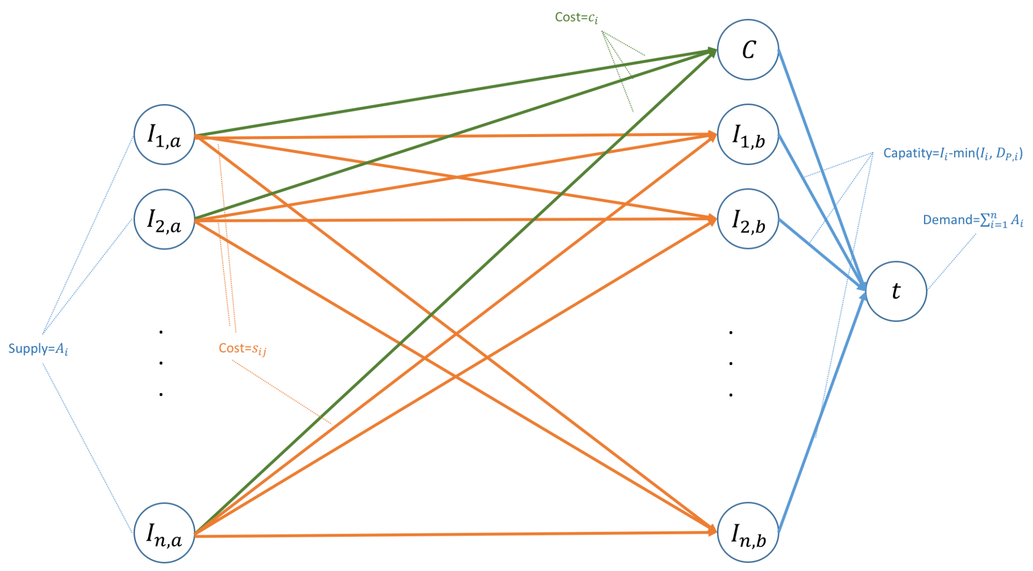


FIGURE 1.5: Single-commodity flow formulation of Linear Program 1.25

that S_k is a variable that can change while all other Local Thresholds S_j for $j \neq k$ are fixed values. We observe that some of the equations change slightly, but the same arguments are true of these revised equations. Now:

$$\sum_{i,j} F_{ij} = \min(\sum_{i=1}^{n} \min(S_i, \mathcal{D}_i^O), \sum_{i=1}^{n}(I_i - \min(I_i, \mathcal{D}_i^P)))$$

and

$$\begin{aligned} &P(S_k, D) \\ =&E_D[p \min(\sum_{i=1}^{n}(I_i - \min(I_i - \mathcal{D}_i^P)) - \min(\sum_{i=1}^{n} \min(S_i, \mathcal{D}_i^O), \sum_{i=1}^{n}(I_i - \min(I_i, \mathcal{D}_i^P))), \\ &\sum_{i=1}^{n}(\mathcal{D}_i^O - \min(S_i, \mathcal{D}_i^O)]. \end{aligned}$$

We can define $D(S_k)$ as the true demand distribution restricted to outcomes where $\mathcal{D}_k^O > S_k$ and we see that $P(S_k+1, D(S_k)) - P(S_k, D(S_k)) = -p \cdot Pr(\sum_{i=1}^{n} \min(I_i, \mathcal{D}_i^P) + \min(S_i, \mathcal{D}_i^O) < \sum_{i=1}^{n} I_i)$. The economic interpretation of the probability in this expression is $-p$ times the probability there is unsold inventory after physical and accepted online orders are filled. This probability is decreasing as we increase S_k and so $P(S_k + 1, D(S_k)) - P(S_k, D(S_k))$ is increasing in S_k. $F(S_k + 1, D(S_k)) -$

$F(S_k, D(S_k))$ is also increasing in S_k as a direct consequence of Lemma 2 by the same argument used in the proof of Theorem 2. It follows that $G(S_k)$ is decreasing at all values of S_k below S_k^* and that $G(S_k)$ is increasing at all values of S_k above S_k^*, concluding the proof.

□

1.6 Complete Retail Network Results

In Section 4, we develop a method that finds optimal local and global threshold policies for our omni-channel fulfillment problem. In this section we will assess the empirical performance of these policies on full-size instances. This will give us insight into the strengths of each policy class while also verifying that this IPA method is of practical use. In our experiments, we will use demand distributions that are estimated from sales and inventory data of an upscale North American retailer. We use the demand data across the full retail network from the top 20 bestselling UPCs at this retailer to generate a realistic instance corresponding to each of these 20 UPCs. This data has been made available to us by analytics firm Onera Inc. A typical instance will have inventory located at 30 to 40 store locations. The cancel parameter is set to two times the price parameter, and ship costs are proportional to distance. Inventories are set to two units at each retail location so as to generate instances where careful supervision of online fulfillment is necessary. For each of these test instances, we compare Local Threshold and Global Threshold policies to Siloed Fulfillment and Reactive Fulfillment policies.

Definition 3. *The Siloed Fulfillment policy treats each store location as a separate retail network and computes the optimal Global Threshold policy for each individual store as its own instance.*

Siloed Fulfillment policies might be used in practice if a retailer is not aware or sophisticated enough to implement a coordinated full-network ship from store program.

Definition 4. *The Reactive Fulfillment policy is the Local Threshold policy that uses the thresholds from the Siloed Fulfillment policy as its threshold parameters.*

Reactive Fulfillment policies use the same set of thresholds for the first stage problem as are computed by the Siloed Fulfillment policy, but the retailer is still able to execute long-distance shipments when solving the second stage problem. A policy similar to the Reactive Fulfillment policy might occur in practice if a retailer sets its online order acceptance thresholds without considering its entire network but has a modern ship from store fulfillment process running in production for its accepted online orders.

We select the 20 UPCs to test our algorithms on by selecting the top 20 UPCs in terms of total sales during the month of May 2016. We used the mean daily number

	Average Cost	Saving %
Siloed Fulfillment	471	-
Reactive Fulfillment	369	21.5%
Global Threshold	116	75.3%
Local Threshold	104	77.9%

FIGURE 1.6: Average fulfillment costs across 20 full-network instances

of units sold at each store location during May 2016, calculated from the retailer's archived analytics data, to estimate Poisson demand distributions at each store locations. This use of these statistics is also consistent with the maximum likelihood estimation approach to estimating Poisson distributions. We compute shipping costs proportional to the Haversine distance between the Zip Codes of every pair of stores in the network. The Haversine distance in kilometers is multiplied by $\frac{1}{250}$ to generate realistic costs. The four fulfillment algorithms are tested in 100 trials for each of the 20 UPCs to generate the results described in this section and Figure 1.6.

We find that across our test instances Local Thresholds and Global Thresholds provide great improvement (78 and 75 percent improvement, respectively) over the Siloed Fulfillment and Reactive Fulfillment policies. For many retailers the Global Threshold policies may be sufficient, providing most of the benefit possible from a network-wide optimization procedure and maintaining a simple policy that can be implemented easily in a production environment. In Section 1.7 we provide additional insight into the tradeoffs between Local Threshold and Global Threshold policies.

1.7 Insights from Two-Store Instances

We conduct experiments on two-store instances of the problem to provide insight into how Local Threshold and Global Threshold policies compare. We are specifically interested in four questions:

1. What is the effect of balanced and imbalanced inventory?
2. How does magnitude of in-store demand affect performance?
3. Does relative performance of policies vary with cancel costs?
4. What conditions result in good performance of Global Thresholds?

To answer each of these questions, we conduct two-store experiments where we run our policies across several instances that vary in a deliberate way across a small number of specific parameters. This controlled variation across instances grants us insight into when and why our methods are effective. To assess the effect of inventory balance, we vary how evenly inventory is distributed between stores, whether this inventory is aligned with demand, and whether the total amount of inventory available modulates with this effect. We investigate the effect of in-store demand magnitude by testing our algorithms on four different in-store demand levels each tested on instances with four different inventory levels. Lastly, we test our algorithms on an instance where cancel cost c is varied from 50% to 400% of the unnecessary rejection penalty p to understand the impact of the ratio of cancel cost to rejection penalty on the relative performance of our methods. We conduct each of these two-store experiments on demand distributions fit to 10 real-life UPCs. This is to make sure our findings are reproducible across various realistic demand patterns.

1.7.1 Inventory Balance

To assess the effect of inventory balance we conducted experiments using distributions fit to 10 popular UPCs. For each UPC, we fit Poisson demand distributions for its two top-selling store locations (with respect to online demand). We allowed inventory to vary at three levels ranging from 50% of mean total demand to 150% of mean total demand. We also let inventory balance vary from 25% to 75% of total inventory in the first location, across three conditions. This results in nine trials for each UPC evaluated. Our primary finding is that Local Thresholds provide the greatest improvement over Global Thresholds when inventory is not aligned with demand. We also observe that this effect is magnified by low inventory levels. As the total amount of starting inventory increases the performances of the two methods become very similar.

We first report overall results in Figure 1.7, averaged by inventory balance condition. Each inventory balance condition was evaluated at 3 total inventory levels per UPC, and 10 UPCs were tested. Every individual instance is evaluated by taking the average cost of each policy over 10000 samples of demand. We call the inventory balance conditions "25%:75%", "50%:50%", and "75%:25%". In these conditions the first percentage refers to the percent of total inventory located at the location with the higher online demand rate, and the second percentage indicates the percent of total inventory located at the location with the lower online demand rate.

Across all instances Local Thresholds slightly outperform Global Thresholds, and both Threshold policies substantially outperform the two benchmark policies, Siloed Fulfillment and Reactive Fulfillment. The gap between our IPA Threshold policies (Local Thresholds and Global Thresholds) and these benchmarks grows in absolute terms yet shrinks in percentage terms as inventory is most out of balance with online demand. Specifically, we see that in the "25%:75%" condition, Local Thresholds average an improvement of 40.7 cost units (25.8%) over Siloed Fulfillment. In the "50%:50%" condition this change becomes 18.2 cost units (28.2%), and in the "75%:25%" condition this change becomes 17.7 cost units (33.3%).

In Figures 1.8-1.10 we split the results presented in Figure 1.7 across the three levels of total inventory tested. We observe that the savings from using Local and Global Threshold Policies are greatest in absolute terms for the middle inventory level, where total inventory available is equal to 100% of the mean total demand.

This demonstrates that the optimization problem studied in this chapter is most difficult when inventory scarcity is at a "sweet spot" such that fulfillment decisions can dramatically influence costs. We also see that Global Thresholds perform poorly when inventory is most scarce, though the performance gap between Local Thresholds and Global Thresholds shrinks as inventory levels increase. At the highest inventory level Local Thresholds and Global Thresholds perform nearly identically – and substantially better than the benchmark policies.

	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%:75%	157.9	145.7	128.0	117.2
50%:50%	64.7	53.2	51.1	46.4
75%:25%	53.1	46.2	40.6	35.4

FIGURE 1.7: Overall average costs of each inventory balance condition for Inventory Balance experiments

	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%:75%	71.5	68.2	90.2	63.1
50%:50%	52.8	45.1	54.3	44.9
75%:25%	39.9	36.0	48.4	36.6

FIGURE 1.8: Average cost of each inventory balance condition for total inventory of 50% mean total demand

	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%:75%	215.9	201.0	168.2	162.8
50%:50%	107.6	90.0	80.8	76.4
75%:25%	74.9	65.4	53.5	49.9

FIGURE 1.9: Average cost of each inventory balance condition for total inventory of 100% mean total demand

	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%:75%	186.2	168.0	125.6	125.7
50%:50%	33.8	24.6	18.1	18.1
75%:25%	44.5	37.2	19.6	19.7

FIGURE 1.10: Average cost of each inventory balance condition for total inventory of 150% mean total demand

Figures 1.11-1.13 plot the results of these experiments for a single UPC, visualizing a characteristic example of the outcomes we observed.

1.7.2 Magnitude of In-Store Demand

To answer this question, we fit online demand to the two top-selling store locations of 10 popular UPCs. We set inventory equal at each location, but we tested four levels of inventory at each location: 5, 10, 15, and 20. For each inventory level we test three Poisson rate parameters of in-store demand: 25%, 50% and 75% of inventory.

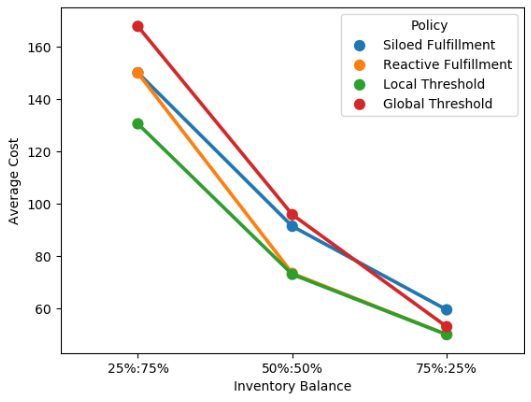


FIGURE 1.11: Average objective value for one UPC as inventory balance is shifted between stores. Inventory is 50% of mean total demand

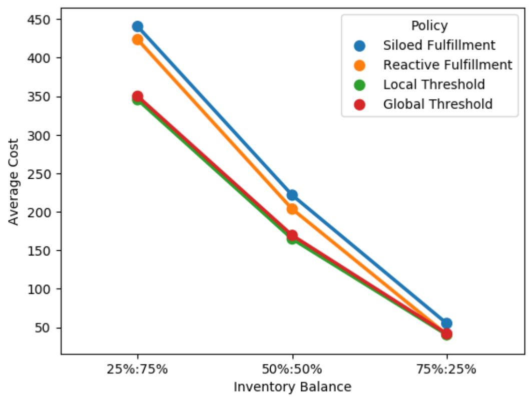


FIGURE 1.12: Average objective value for one UPC as inventory balance is shifted between stores. Inventory is 100% of mean total demand

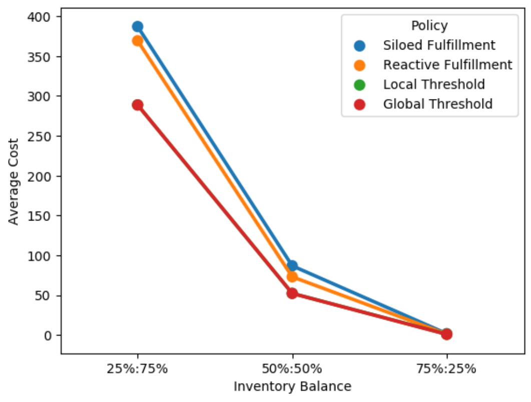


FIGURE 1.13: Average objective value for one UPC as inventory balance is shifted between stores. Inventory is 150% of mean total demand

This results in 12 trials for each UPC. We observe that Local Thresholds outperform Global Thresholds across all scenarios, but the performance of Local Threshold policies is more sensitive to increases in in-store demand. For example, increasing mean in-store demand from 25% to 50% of the inventory level increased Local Threshold costs by an average of 5.4% compared to only a 0.9% average cost increase for Global Threshold policies. This difference in sensitivity to changes of in-store demand rates can be seen consistently across the scenarios evaluated. These results are presented in Figures 1.14-1.18. In our first trial, where inventory is set to 5 units at both store locations, Global Thresholds comes closest to the performance of Local Thresholds when in-store demand is high. In our other trials with higher inventory levels Local Thresholds outperform Global Thresholds when in-store demand is low and Global Thresholds perform similar to Local Thresholds when in-store demand is high. We expect that when inventory is low and in-store demand is high, both Global Thresholds and Local Thresholds will accept a similar set of orders. Local Thresholds policies still have a small advantage, though, because they are able to ensure that accepted orders are balanced between stores. In Section 1.7.4 we consider related scenarios involving correlated demands that could give Global Thresholds an advantage of Local Thresholds.

1.7.3 Impact of Cancel Costs

In this experiment, we fit demand to two top-selling store locations of 10 popular UPCs. We set test three inventory levels: 5, 10, and 15 units at each store location. For each of these inventory levels we compare our policies at the following cancel costs: 20, 40, 60, and 80. This results in 12 total trial for each UPC. The price of the item is set to 20 for all trials. Results from these experiments are presented in Figures 1.19-1.22.

Our first finding is that the performance of Global Thresholds is less sensitive to changes in cancel cost than are Local Thresholds. We observe that as cancel costs increase, Local Thresholds' performance decreases both in absolute terms and in comparison to Global Thresholds. At the lowest cancel penalty, 20, Local Thresholds incur 13% lower costs on average than Global Thresholds. For our trials with cancel penalty 80 this decrease in costs is on 9% on average. A likely explanation for these results is that any advantage Local Threshold polices have over Global Threshold policies comes from their ability to balance accepted orders across store locations. When the cancel penalty is high, the total amount of cancel costs realized becomes a more important factor that is better managed by Global Thresholds.

We also find that the performance differences between Global Thresholds and Local Thresholds are magnified at higher inventory levels. The percentage decrease in average cost from Global Thresholds to Local Thresholds ranged from -3% to 3% at inventory level 5. At inventory level 15, these same average percentage decreases in cost ranged from 13% to 16%. When inventory levels are low the policies will accept fewer orders, which can result in the Local Threshold and Global Threshold

Demand	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%	54.4	47.0	50.8	41.7
50%	58.8	47.1	51.2	43.9
75%	63.1	47.9	52.5	46.9

FIGURE 1.14: Overall average costs of each in-store demand condition for demand magnitude experiments

Demand	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%	30.1	23.5	31.8	23.3
50%	40.13	47.1	34.2	30.1
75%	42.1	47.9	33.6	32.2

FIGURE 1.15: Average cost of each in-store demand condition for starting inventory of 5 at each location

Demand	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%	48.4	40.9	47.8	37.1
50%	52.6	40.5	47.2	38.9
75%	59.6	45.1	53.9	44.3

FIGURE 1.16: Average cost of each in-store demand condition for starting inventory of 10 at each location

Demand	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%	66.5	59.6	61.2	51.7
50%	65.6	53.8	57.1	48.6
75%	70.9	53.1	58.5	52.3

FIGURE 1.17: Average cost of each in-store demand condition for starting inventory of 15 at each location

Demand	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
25%	72.6	64.1	62.2	54.6
50%	76.9	63.6	66.4	58.1
75%	79.8	59.8	64.1	58.6

FIGURE 1.18: Average cost of each in-store demand condition for starting inventory of 20 at each location

Cancel Penalty	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
20	60.6	50.9	52.5	45.7
40	71.3	60.1	58.8	53.1
60	77.4	66.4	62.7	57.1
80	81.8	71.1	65.8	60.0

FIGURE 1.19: Overall average costs of each cancel penalty condition for cancel cost magnitude experiments

Cancel Penalty	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
20	29.0	25.8	25.9	25.1
40	32.0	29.0	28.4	27.8
60	33.9	30.2	28.9	29.9
80	35.7	31.1	29.7	29.9

FIGURE 1.20: Average costs of each cancel penalty condition for starting inventory of 5 at each location

Cancel Penalty	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
20	65.7	53.3	57.3	49.6
40	80.7	65.0	66.0	61.1
60	88.8	76.1	71.8	65.6
80	94.0	81.3	76.7	70.8

FIGURE 1.21: Average costs of each cancel penalty condition for starting inventory of 10 at each location

Cancel Penalty	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
20	87.2	73.8	74.2	62.5
40	101.1	86.2	81.9	70.5
60	109.5	92.9	87.5	75.7
80	115.6	100.9	91.1	79.2

FIGURE 1.22: Average costs of each cancel penalty condition for starting inventory of 15 at each location

policies accepting very similar sets of orders, reducing the observable differences in performance between these policies.

1.7.4 Global Thresholds Performance

Throughout our previous experiments we have found Local Thresholds to consistently outperform Global Thresholds, though often by only a small margin. The instances tested in these experiments are formulated from distributions that were chosen to be similar to what we observe in real retail data, so a reasonable conclusion may be that Local Thresholds are a high-performing policy for real-life scenarios. However, in this section we explore potentially artificial scenarios that result in Global Threshold policies outperforming Local Threshold policies.

In-Store Var	Online ρ	Siloed Fulfill	Reactive Fulfill	Global Thresh	Local Thresh
1.5	-.7	41.2	26.6	16.5	22.8
6	-.7	61.2	36.8	30.7	34.2
10.5	-.7	74.1	42.4	37.2	40.4
1.5	0	32.1	20.0	15.5	19.9
6	0	56.0	31.8	28.9	31.8
10.5	0	67.9	38.5	37.1	38.5
1.5	.7	25.3	15.9	15.5	15.9
6	.7	51.1	31.2	29.1	29.6
10.5	.7	65.6	38.2	36.4	36.7

FIGURE 1.23: Average cost of each covariance condition

We consider a set of two-store instances where inventory is fixed at 20 at both locations. $c = p = 20$, and the shipping cost between the two stores is $.5$. Demand distributions are multivariate normal (rounded to the nearest non-negative integer), and we will vary the covariance matrix across several conditions. The in-store demand distribution has mean demand 15 at each location and values along the diagonal of the covariance matrix 1.5, 6, and 10.5 across three conditions tested. The covariance of in-store demand between the two locations is zero. The online demand distribution has mean demand 5 at each location and the values along the diagonal of the covariance matrix are 5. The covariance between the two stores is set such that the correlation coefficient of online demand is $-.7$, 0, and $.7$ across three conditions tested. We evaluated 10000 observations of demand across all combinations of the three in-store demand conditions and the three online demand conditions, resulting in 9 total conditions tested. Full results from these trials are displayed in Figure 1.23.

There are several factors that influence the results of these trials in favor of the Global Threshold policies. For one, we set shipping costs to be relatively low. If shipping costs are zero then the model becomes equivalent to a single-store instance and Local Threshold will no longer outperform Global Threshold, but this alone is not always enough to give Global Threshold a distinct advantage. In particular, for Global Threshold to have an advantage online demand rates should be in a narrow range where there is enough demand to differentiate the policies but not enough demand for a Local Threshold policy to accept always accept up to its threshold value at all locations. This intuition led us to set the mean online demand to 5 at each location. We verify this intuition by varying the mean online demand from 1 to 15 at each location for the first condition tested in this set of trials, where in-store variance is 1.5 and online correlation is -.7. The average costs at each demand rate are plotted in Figure 1.24, confirming that the advantage Global Threshold policies have in this covariance scenario are only visible through a somewhat narrow range of demand distributions.

We hypothesized that negatively correlated online demand would further help Global Threshold policies. Negatively correlated online demand will require Local Threshold policies to be set at fairly high levels for both store locations. This is

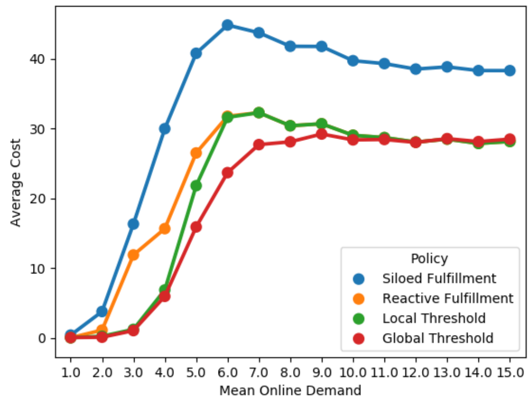


FIGURE 1.24: Average objective value as the mean online demand parameter is shifted

because most of the time the demand will be imbalanced between the stores. Occasionally there will still be high demand at both locations and this will cause the Local Threshold policy to accept more orders than is ideal. Global Threshold policies are largely unaffected by correlation in online demand, though they may end up paying higher shipping costs than a Local Threshold policy especially if online demand is negatively correlated. Our results are consistent with this observation, where Global Threshold policies have 19% lower cost than Local Threshold policies when online demand has a correlation coefficient of -.7, compared to cost decreases of 7% and 4% for correlation coefficients 0 and .7, respectively.

Similarly, we expected lower rates of in-store variance to benefit Global Thresholds. We expected Global Threshold policies to outperform Local Threshold policies when total online demand tends to be relatively close to the total available inventory after fulfilling in-store demand. Our results along this dimension are also consistent with this hypothesis. Global Threshold policies have 13% lower cost than Local Threshold policies when in-store variance is 1.5, compared to cost decreases of 10% and 1% for variances of 6 and 10.5, respectively.

1.8 Extensions

1.8.1 Nested Threshold Policies

Nested Threshold policies are a class of policies that generalize Local Thresholds and Global Thresholds by implementing a global threshold across all stores in addition to a local threshold at each store location. Nested Thresholds are easy to implement as all threshold parameters can be updated using the same IPA methods and gradient estimates developed to update Local Threshold and Global Threshold policies separately. We evaluated Nested Thresholds in the experiments conducted in Sections 1.6 and 1.7, and we found that in nearly all cases Nested Thresholds closely tracked either Local Thresholds or Global Thresholds, whichever method was performing better on that instance. An interesting future direction could be to seek out conditions where Nested Threshold policies strictly dominate Local Thresholds and Global Thresholds or to argue why such conditions could not exist.

1.8.2 Single Store Instances with Unknown Demand Distributions

In this subsection, we consider the same model as in Section 1.4, except now the demand distributions are unknown to the retailer. Instead, the retailer is given N i.i.d. samples from the demand distributions for online and in-store demand. We assume in this section that in-store and online demands are independent. In the case of nonzero correlation between in-store and online demand, comparable bounds to Theorem 4 can be proven by taking two samples of N observations ($2N$ total observations) and using one set of N observations to estimate the empirical distribution of $\mathcal{D}^O$ and the other set of N observations to estimate the empirical distribution of $\mathcal{D}^P$.

I demonstrate that for sufficiently large values of N, the retailer can use the observed empirical distribution of the N samples as an approximation of the true demand distributions and set a policy whose performance is arbitrarily close to that of the optimal policy. Let $\hat{F}_P()$ to be the empirical distribution function of observed physical demand and let $\hat{S} = I - \hat{F}_P^{-1}(\frac{c}{c+p})$.

Definition 5. *Let S be a constant and let $0 < \alpha < 1$. We will say that S is α-accurate if $F_P(I - S) \geq \frac{c}{c+p} - \alpha$, $\bar{F}_P(I - S) \geq \frac{p}{c+p} - \alpha$.*

This definition is a modified version of the definition of α-accuracy in [42]. We use tail bounds to reason about the probability $\hat{S}$ is α-accurate when computed from a sample of n observations:

Lemma 3. *Threshold $\hat{S} = I - \hat{F}_P^{-1}(\frac{c}{c+p})$, generated from a sample of N observations of $\mathcal{D}^P$, is α-accurate with probability at least $1 - 2\epsilon^{-2N\alpha^2}$.*

Proof. The empirical CDF from N independent random samples of physical demand $(d_1^P, \ldots, d_N^P)$ can be expressed as the sum of N independent $[0,1]$ variables:

$$\hat{F}_P(x) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[d_i^P \leq x].$$

The expected value of each of these terms is equal to the true CDF evaluated at x, $E[\mathbf{1}[d_i^P \leq x]] = F_P(x)$, and so $E[\hat{F}_P(x)] = F_P(x)$. Then, we can use Hoeffding's

	$d^O \leq \hat{S}$	$\hat{S} < d^O < S^*$	$S^* \leq d^O$
$d^P \leq I - \hat{S}$	0	$p(d^O - \hat{S})$	$p(S^* - \hat{S})$
$I - \hat{S} \leq d^P$	0	$-c(d^O - \hat{S})$	$-c(S^* - \hat{S})$

FIGURE 1.25: Upper bounds on $g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P)$ when $\hat{S} \leq S^*$

Inequality [45] to bound the probability the empirical CDF is close to the true PDF:

$$\begin{aligned}
&\Pr(|\frac{c}{c+p} - F_P(I - \hat{S})| \geq \alpha) \\
=&\Pr(|\hat{F_P}(I - \hat{S}) - F_P(I - \hat{S})| \geq \alpha) \\
=&\Pr(|\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[d_i^P \leq I - \hat{S}] - E[\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[d_i^P \leq I - \hat{S}]]| \geq \alpha) \\
\leq&2e^{-2N\alpha^2}.
\end{aligned}$$

□

Lemma 4. *Suppose $\hat{S}$ is α-accurate, then $G(\hat{S}) \leq (1+\epsilon)G(S^*)$ for $\epsilon \geq \frac{\bar{F}_O(\hat{S})\alpha(p+c)}{\bar{F}_O(S^*)(\frac{cp}{c+p} - \alpha c)}$*

Proof. First, we will prove this result for the case where $\hat{S} < S^*$. We will then use a similar argument to consider the remaining case, though it this first case that requires a stronger condition on ϵ.

Let $g(S, d^O, d^P)$ represent the cost function for threshold S and realized demands d^O and d^P. Figure 1 presents upper bounds on $g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P)$ for all possible values of $\mathcal{D}^O$ and $\mathcal{D}^P$, broken into cases:

In this case,

$$G(\hat{S}) - G(S^*) \leq F_O(\hat{S})0 + \Delta_2 + \Delta_3$$

where $\Delta_2 = E_{d^O,d^P}[g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P)|\hat{S} < d^O < S^*]$ and $\Delta_3 = E_{d^O,d^P}[g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P)|d^O \geq S^*]$

$$\begin{aligned}
\Delta_2 = \quad & \int_{\hat{S}}^{S^*} E_{\mathcal{D}^O,\mathcal{D}^P}[g(\hat{S},\mathcal{D}^O,\mathcal{D}^P) - g(S^*,\mathcal{D}^O,\mathcal{D}^P)|\mathcal{D}^O = x]f_O(x)dx \\
= \quad & \int_{\hat{S}}^{S^*} \int_0^{\infty} [g(\hat{S},x,y) - g(S^*,x,y)]f_O(x)f_P(y)dydx \\
= \quad & \int_{\hat{S}}^{S^*} [\int_0^{I-\hat{S}} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy \\
& + \int_{I-\hat{S}}^{\infty} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy]dx \\
\leq \quad & \int_{\hat{S}}^{S^*} [F_P(I-\hat{S})p(x-\hat{S}) - (\bar{F}_P(I-\hat{S})c(x-\hat{S})]dx \\
\leq \quad & \int_{\hat{S}}^{S^*} [(\frac{c}{c+p} + \alpha)(p+c)(x-\hat{S}) - (c(x-\hat{S})]dx \\
\leq \quad & \int_{\hat{S}}^{S^*} [\alpha(p+c)(S^*-\hat{S})]dx \\
= \quad & (F_O(S^*) - F_O(\hat{S}))[\alpha(p+c)(S^*-\hat{S})]
\end{aligned}$$

and

$$\begin{aligned}
\Delta_3 = \quad & \int_{S^*}^{\infty} E_{\mathcal{D}^O,\mathcal{D}^P}[g(\hat{S},\mathcal{D}^O,\mathcal{D}^P) - g(S^*,\mathcal{D}^O,\mathcal{D}^P)|\mathcal{D}^O = x]f_O(x)dx \\
= \quad & \int_{S^*}^{\infty} \int_0^{\infty} [g(\hat{S},x,y) - g(S^*,x,y)]f_O(x)f_P(y)dydx \\
= \quad & \int_{S^*}^{\infty} [\int_0^{I-\hat{S}} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy \\
& + \int_{I-\hat{S}}^{\infty} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy]dx \\
\leq \quad & \int_{S^*}^{\infty} [F_P(I-\hat{S})p(S^*-\hat{S}) - (\bar{F}_P(I \hat{-} S)c(S^*-\hat{S}))]dx \\
\leq \quad & \int_{S^*}^{\infty} [(\frac{c}{c+p} + \alpha)(p+c)(S^*-\hat{S}) - (c(S^*-\hat{S}))]dx \\
\leq \quad & \int_{S^*}^{\infty} [\alpha(p+c)(S^*-\hat{S})]dx \\
= \quad & \bar{F}_O(S^*)[\alpha(p+c)(S^*-\hat{S})].
\end{aligned}$$

Then, $G(\hat{S}) - G(S^*) \leq \bar{F}_O(\hat{S})[\alpha(p+c)(S^*-\hat{S})]$.
Let $\Delta(\hat{S}) = \alpha(p+c)(S^*-\hat{S})$. Then

$$\begin{aligned}
G(\hat{S}) - G(S^*) \leq \quad & \bar{F}_O(\hat{S})\Delta(\hat{S}) \\
= \quad & \bar{F}_O(\hat{S})\alpha(p+c)(S^*-\hat{S}).
\end{aligned}$$

	$d^O \le S^*$	$S^* < d^O < \hat{S}$	$\hat{S} \le d^O$
$d^P \le I - \hat{S}$	0	$-p(d^O - S^*)$	$-p(\hat{S} - S^*)$
$I - \hat{S} \le d^P$	0	$c(d^O - S^*)$	$c(\hat{S} - S^*)$

FIGURE 1.26: Upper bounds on $g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P)$ when $\hat{S} > S^*$

We also lower bound $G(S^*)$:

$$\begin{aligned}
G(S^*) \ge & \quad \bar{F}_O(S^*)\bar{F}_P(I - \hat{S})c(S^* - \hat{S}) \\
\ge & \quad \bar{F}_O(S^*)(\frac{p}{c+p} - \alpha)c(S^* - \hat{S}) \\
= & \quad \bar{F}_O(S^*)(\frac{cp}{c+p}(S^* - \hat{S}) - \alpha c(S^* - \hat{S})) \\
= & \quad \bar{F}_O(S^*)(\frac{cp}{c+p} - \alpha c)(S^* - \hat{S}).
\end{aligned}$$

Consequently, the following is a sufficient condition for an ϵ-optimal solution when $\hat{S} \le S^*$.

$$\epsilon \ge \frac{\bar{F}_O(\hat{S})\alpha(p+c)}{\bar{F}_O(S^*)(\frac{cp}{c+p} - \alpha c)}.$$

This is the required criteria in the Lemma's statement, and so the Lemma is valid for this case when $\hat{S} \le S^*$

Next, we perform a similar analysis for the remaining case, where $\hat{S} > S^*$, from which we will obtain a slightly relaxed version of the above condition on ϵ.

In this case,

$$G(\hat{S}) - G(S^*) \le F_O(S^*)0 + \Delta_2 + \Delta_3$$

where $\Delta_2 = E_{d^O,d^P}[g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P) | S^* < d^O < \hat{S}]$ and $\Delta_3 = E_{d^O,d^P}[g(\hat{S}, d^O, d^P) - g(S^*, d^O, d^P) | d^O \ge \hat{S}]$

$$\begin{aligned}
\Delta_2 = \quad & \int_{S^*}^{\hat{S}} E_{\mathcal{D}^O,\mathcal{D}^P}[g(\hat{S},\mathcal{D}^O,\mathcal{D}^P) - g(S^*,\mathcal{D}^O,\mathcal{D}^P)|\mathcal{D}^O = x]f_O(x)dx \\
= \quad & \int_{S^*}^{\hat{S}} \int_0^{\infty} [g(\hat{S},x,y) - g(S^*,x,y)]f_O(x)f_P(y)dydx \\
= \quad & \int_{S^*}^{\hat{S}} [\int_0^{I-\hat{S}} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy \\
& + \int_{I-\hat{S}}^{\infty} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy]dx \\
\leq \quad & \int_{S^*}^{\hat{S}} [-F_P(I-\hat{S})p(x-S^*) + (\bar{F}_P(I-\hat{S})c(x-S^*)]dx \\
\leq \quad & \int_{S^*}^{\hat{S}} [(c(x-S^*) - (\frac{c}{c+p} - \alpha)(p+c)(x-S^*)]dx \\
\leq \quad & \int_{S^*}^{\hat{S}} [\alpha(p+c)(\hat{S}-S^*)]dx \\
= \quad & (F_O(\hat{S}) - F_O(S^*))[\alpha(p+c)(\hat{S}-S^*)]
\end{aligned}$$

and

$$\begin{aligned}
\Delta_3 = \quad & \int_{\hat{S}}^{\infty} E_{\mathcal{D}^O,\mathcal{D}^P}[g(\hat{S},\mathcal{D}^O,\mathcal{D}^P) - g(S^*,\mathcal{D}^O,\mathcal{D}^P)|\mathcal{D}^O = x]f_O(x)dx \\
= \quad & \int_{\hat{S}}^{\infty} \int_0^{\infty} [g(\hat{S},x,y) - g(S^*,x,y)]f_O(x)f_P(y)dydx \\
= \quad & \int_{\hat{S}}^{\infty} [\int_0^{I-\hat{S}} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy \\
& + \int_{I-\hat{S}}^{\infty} (g(\hat{S},x,y) - g(S^*,x,y))f_O(x)f_P(y)dy]dx \\
\leq \quad & \int_{\hat{S}}^{\infty} [-F_P(I-\hat{S})p(\hat{S}-S^*) + (\bar{F}_P(I \hat{-} S)c(\hat{S}-S^*))]dx \\
\leq \quad & \int_{\hat{S}}^{\infty} [(c(\hat{S}-S^*)) - (\frac{c}{c+p} - \alpha)(p+c)(\hat{S}-S^*)]dx \\
\leq \quad & \int_{\hat{S}}^{\infty} [\alpha(p+c)(\hat{S}-S^*)]dx \\
= \quad & \bar{F}_O(\hat{S})[\alpha(p+c)(\hat{S}-S^*)].
\end{aligned}$$

Then, $G(\hat{S}) - G(S^*) \leq \Delta_2 + \Delta_3 \leq \bar{F}_O(S^*)[\alpha(p+c)(\hat{S}-S^*)]$.
Let $\Delta(\hat{S}) = \alpha(p+c)(\hat{S}-S^*)$. Then

$$\begin{aligned}
G(\hat{S}) - G(S^*) \leq \quad & \bar{F}_O(S^*)\Delta(\hat{S}) \\
= \quad & \bar{F}_O(S^*)\alpha(p+c)(\hat{S}-S^*).
\end{aligned}$$

We also lower bound $G(S^*)$:

$$\begin{aligned} G(S^*) \geq & \quad \bar{F}_O(S^*)\bar{F}_P(I-\hat{S})c(\hat{S}-S^*) \\ \geq & \quad \bar{F}_O(S^*)(\frac{p}{c+p}-\alpha)c(\hat{S}-S^*) \\ = & \quad \bar{F}_O(S^*)(\frac{cp}{c+p}(S^*-\hat{S})-\alpha c(\hat{S}-S^*)) \\ = & \quad \bar{F}_O(S^*)(\frac{cp}{c+p}-\alpha c)(\hat{S}-S^*). \end{aligned}$$

Consequently, the following is a sufficient condition for an ϵ-optimal solution when $\hat{S} > S^*$:

$$\epsilon \geq \frac{\alpha(p+c)}{(\frac{cp}{c+p}-\alpha c)}.$$

This is a relaxation of the required criteria in the Lemma's statement and therefore the Lemma also holds when $\hat{S} > S^*$, concluding the proof. ☐

Theorem 4. *For each $\epsilon > 0$ and $0 < \delta < 1$, if the number of samples, N meets criteria*

$$N \geq \frac{-\log(\frac{\delta}{2})}{2\log(\epsilon)}(\frac{\bar{F}_O(\hat{S})(p+c)+\epsilon\bar{F}_O(S^*)c}{\epsilon\bar{F}_O(S^*)\frac{cp}{c+p}})^2$$

then $G(\hat{S}) \leq (1+\epsilon)G(S^*)$ with probability at least $1-\delta$.

Proof. Theorem 4 follows directly as a consequence of Lemma 3 and Lemma 4. ☐

We observe from Theorem 4 that the sample size requirement for Sample Average Approximation to produce a near-optimal solution at a high probability depends on the demand distributions only with respect to the convergence rate of the empirical quantile function of online demand. Exact convergence rates of the empirical quantile function to the true quantile function for general probability distributions are not known, but some asymptotic properties are known in the statistics literature (see [54] for the proof of Lemma 5 and additional information on the convergence of the empirical quantile distribution):

Lemma 5. *Fix $0 < p < 1$. If CDF F if differentiable at $F^{-1}(p)$ with positive derivative $f(F^{-1}(p))$, then $\sqrt{n}(F_N^{-1}(p)-F^{-1}(p))$ is asymptotically normal with mean 0 and variance $\frac{p(1-p)}{f^2(F^{-1}(p))}$.*

Proof of Lemma 5 can be found in [54].

1.9 Conclusion

In this chapter we introduce a new stochastic model for omni-channel fulfillment. This model incorporates new risks that occur when fulfillment operations are combined for in-store and online demand. We obtain a closed-form optimal solution to this model for instances with only a single store location. We also identify some structural similarities between this model and the classical newsvendor model that inform our understanding of this model even in multiple store settings.

We continue by studying the complete, multiple-store setting of this model, where we introduce Local Threshold and Global Threshold policy classes for the first stage problem. We also present a sampling-based Infinitesimal Perturbation Analysis algorithm to optimize threshold policies within each of these policy classes.

Next, we evaluate our methods on a variety of test instances. At first, we attempt to create realistic instances and find that our IPA-optimized Local Threshold policies consistently outperform Global Threshold policies and other benchmark policies on these instances. We also explore several environmental factors and discuss how changes along these dimensions affects the performance of our policies.

We continue in the next chapter to investigate policies for omni-channel fulfillment, shifting our focus toward limiting order cancellations globally across an entire catalog of products sold by a retailer. We use techniques from machine learning and discrete optimization to produce store-wide policies that maximize revenues while limiting cancellations in a coordinated manner.

Acknowledgments: Prof. R. Ravi and Prof. Sridhar Tayur are co-authors on this work. We thank Dr. Srinath Sridhar for access to real retail data and helpful discussions that informed this work.

Chapter 2

Optimizing Inventory Exposure with Knapsack Threshold Models

2.1 Introduction

One of the most recent revolutions in e-commerce is *omni-channel fulfillment*. The idea of omni-channel fulfillment is to merge the multiple sales channels of the retailer into one seamless experience and present a unified view of inventory. In addition to improving customer experience, omni-channel retailing provides an opportunity to realize new operational efficiencies by optimizing across retail channels rather than treating them as separate silos. The impact of omni-channel fulfillment has been so immense that it is now routinely cited in quarterly earnings reports as the primary driver in the increased top line sales and/or reduction in costs [37, 25, 3, 18].

SFS and BOPUS. In this chapter, we focus on omni-channel *ship-from-store* (SFS) programs where a customer can buy an item online and the merchant fulfills that order by shipping from a store as opposed to shipping from a traditional warehouse. There are several advantages of SFS initiatives. Firstly, the volume of inventory sold in stores is an order of magnitude higher than the volume of inventory sold on the retailers websites. Therefore at any point in time, the combined network-wide store inventory is significantly higher than warehouse inventory from where online orders are traditionally fulfilled. Other advantages of SFS programs include faster fulfillment time, cheaper fulfillment cost and possible avoidance of store markdowns. This has triggered many of the top retailers including Best Buy, Gap, Macy's, Sears, Toys R' Us just to name a select few, to turn on omni-channel initiatives such as SFS and BOPUS [49, 50]. The models of this chapter also extend in a straightforward way to omni-channel buy-online-pickup-in-stores (BOPUS) programs.

Challenges. Stores are designed for store fulfillment where a customer walks into a store and buys an item. Similarly warehouses are designed for online fulfillment where items are carefully categorized and meticulously organized spatially and operational efficiencies are pushed to the limits. The biggest obstacle that retailers face while enabling SFS programs is *inventory accuracy* in stores. While warehouse inventory is extremely accurate, store inventory is highly inaccurate because of shrinkage, misplacement and inventory cycle counts that are only performed a few times a year. As an example, [15] shows that a multi-billion dollar retailer has roughly 65% of their store SKUs with incorrect inventory.

Balancing Revenue with Cancellations. When SFS programs are enabled by top retailers, they can not take on too many online orders relying on the store inventory systems and later cancel those orders because they can not be fulfilled. It is unacceptable to have customer experience suffer significantly while the top line revenue grows in the short term. This leads to an interesting trade-off between maximizing top-line revenue by making store inventory available online for SFS programs and ensuring that customer experience does not suffer because of too many cancels.

Threshold Policies. Retailers that enable SFS programs therefore adopt strategies to ensure that orders do not cancel because of inaccurate store inventory. These are typically threshold policies that just expose those items that have more than t units network wide across all stores. For example, if $t = 5$, this would simply only make items that have more than 5 units across all stores to be made available for SFS. The idea is simple and intuitive that the inventory systems have to be so inaccurate that 5 is in reality 0 for the order to be canceled. The effect of such simplistic models is significant on revenue. As an example, most pieces of expensive jewelry or cameras may be made unavailable for SFS programs. In fact, the long tail nature of the inventory counts would make such strategies suppress most products from being made available for SFS. Onera Inc., an analytics firm that provides decision support for omni-channel retail, has designed algorithms that model each product differently based on the product ontology, price, price status and the inventory count. As examples, it knows to treat diamond jewelry, scarves, socks, gift cards, and clearance bin items differently from both standpoints of inventory accuracy as well as expected top-line revenue. This optimization problem is one of Onera's core product offerings. The work presented in Chapters 2 and 3 of this thesis was done in collaboration with Dr. Srinath Sridhar of Onera to thoroughly study this very interesting problem.

Contributions. We introduce the problem of optimizing inventory for omni-channel fulfillment. We build a practical model that shows how top-line revenue can be maximized while customer experience is kept within satisfactory limits and the inventory assortment made available online is broad. Although our model suggests an NP-complete formulation, we show that it can be solved to near optimality in both theory and practice. In this chapter we describe and investigate models which first estimates a data generation model and then feed these estimates to an integer programming optimization model to determine thresholds. We call these Separate Estimation Optimization (SEO) models. Chapter 3 studies a second class of models for this problem. These models are end-to-end optimizers that jointly estimate the model parameters as well as the threshold decisions, and are termed Joint Estimation Optimization (JEO) models. We use real-life data to fit realistic generative models to capture demands, inventory positions and cancel rates. An important practical concern is that the real-life training data used for these models have an intrinsic data truncation problem (censoring) which we describe how to address. We assess the accuracy of the coefficients estimated, the recovery of truncated data and the performance of our model on revenues and cancellations through experimental trials using real data from a major online retailer. These experiments provide a taste for the massive impact our models can have on the top-line revenue of a multi-billion dollar online retailer.

2.2 Literature Review

Retail Applications of Machine Learning. The value of our optimization models rely on applying machine learning techniques to structured retail data. Tree-based learning algorithms have been successfully used at Rue La La to generate inputs for an optimization model of pricing decisions [17]. Forecasting and optimization methods have been combined [19, 22] for a method to make shipment decisions at Zara. Large scale network flow models have been used make similar pricing decisions [33].

Learning from Truncated and Censored Data. Learning and optimization based on truncated (or censored) data is an important element of our work, and we benefit from insights previously made in this area. We use the EM algorithm, a well celebrated classical method [16], as a tool to counter data truncation. EM methods are used in a similar way [10] in the context of analyzing medical data, and to solve a demand estimation problem with incomplete data [6] among many other applications. Bayesian methods have also been used to solve Operations Management problems involving censored data [12, 44]. Survival models have also used successfully [55, 4] to learn from censored data in the context of online advertising.

Omni-Channel Retailing. Although the problem of managing cancellations in omni-channel retailing is new, researchers are studying other challenges related to omni-channel retailing using game-theoretic models of customer-firm interactions [20, 21] Lastly, the idea of using thresholds to limit cancellations is conceptually similar to that of safety stocks, a class of policies commonly used in Operations Management on models such as the Newsvendor problem [47] and the transshipment problem [31].

Optimization. One of the pure optimization based methods we use is Sample Average Approximation (SAA), a method for stochastic optimization [39]. We also note that an approximation algorithm for a nonlinear knapsack [32] can be applied to efficiently solve our optimization models, which can also be viewed as a multi-objective optimization model of the ϵ-Constraint Method [14]. The latter has been applied along with machine learning methods in areas such as email volume optimization [28] and recommendation systems [1].

2.3 Problem Formulation

At a high level, the problem of optimizing inventory for omni-channel fulfillment is to select the set of store products to make available online using threshold policies. Such policies assign each product an integer threshold that is passed to the retailer's inventory system. Each product is then made available for SFS if its (network-wide) inventory level is at or above its threshold. The idea is that the thresholds need to be updated only periodically and if the inventory gets low as units are sold, the items would automatically be suppressed from online availability once they fall below the threshold. This allows retailers to automate and optimize a process that is often otherwise performed in a manual and ad-hoc manner. Our models assign thresholds to items based on a variety of features of the product including its price, price status (regular price, markdown, clearance etc.), its location in the hierarchy of the product ontology and the historical cancellation behavior.

2.3.1 Threshold Choice Models and Truncation

We model the problem of setting optimal thresholds as a variant of the classical knapsack problem. In our setting, we have I products and the goal is to find thresholds t_i for every product i. The intent is that item i will be made available to be purchased online via SFS if and only if the network-wide inventory across all stores is at least t_i.

Our model is set up to optimally trade off between multiple objectives such as cancellations, revenues and orders accepted. For threshold t_i, we define expected revenue and cancellations as functions of our thresholds: $r(t_i)$, and $c(t_i)$. The objective is to find t_i for all items i so as to maximize the total expected revenue $\sum_{i=1}^{I} r_i(t_i)$ subject to the expected total items canceled being at most C, i.e. $\sum_{i=1}^{I} c_i(t_i) \leq C$. Note that if $t_i = 0$ for all i, there will be maximum revenue but there will likely also be a lot of cancels. Intuitively this is because the orders accepted at very low inventory counts are likely to cancel. If $t_i = \infty$ for all i, then the SFS program is void and no orders will be accepted.

Truncated demand data. An important aspect in our model is that coefficients $c_i(t_i)$ and $r_i(t_i)$ must be learned from data. Specifically using historical data, we need to estimate for each product i, the expected cancellations and revenue if we set a threshold of t_i. These expectations are over the next period over which the thresholds will be active, which for the purpose of this chapter is the next day. However, intrinsically historical data will only contain online orders for items with inventory counts greater than their thresholds that were set before. Thus, we will not observe any demand or any cancels for any items i where inventory counts were lower than the historical thresholds t_i. Therefore to obtain coefficients $c_i(t_i)$ and $r_i(t_i)$ when inventory is less than t_i, we have to learn based on the orders and cancellations at higher inventory counts that we do observe among other similar products. We consider several methods for estimating these coefficients in Sections 2.6 and 3.5.

2.3.2 Data Sources

To solve this optimization problem, Onera receives feeds that are in three parts: Orders, Inventory and Products. Note that, these are received typically hourly, but for the purpose of this chapter, we can assume that the models are constructed every day and output of the models, the thresholds t_i for different items (either SKUs of if the data is too sparse for categories of SKUs), are updated by the retailer for the next day. We summarize the information received in each of the feeds below leaving some details for the sake of simplicity.
Orders: We can express this as a tuple of (order id, item id, status). An order id is a unique id corresponding to a customer's order, the item id is a unique id corresponding to the item that was ordered and the binary status can be one of $\{0, 1\}$ indicating if the order was canceled or fulfilled respectively[1]. We will assume that every order is either fulfilled or canceled wholly and a cancellation occurs only due to inventory not being found in the store when the inventory system indicated otherwise[2].
Inventory: This feed is a snapshot in time of the inventory across all stores and can be expressed as the following tuple: (item id, inventory count).
Products: This feed describes the items that are being sold by the retailer so that one can analyze their ontology to generate features that help with our parameter estimations. This feed is keyed by the item id, contains the location in the product ontology (typically a 4th to 7th level tree node in the product hierarchy), the price, and price status (such as regular, markdown, clearance etc). The price, for the sake of the chapter, will refer to the online price although there is a distinction that can be made between the online price, per store price and the price that the customer actually paid for the order.
Cancel Constraint: This is simply the number C specified in the above model that is a business constant provided by the retailer to Onera. The idea is to try to ensure that cancels do not exceed C. Note that C can be assumed to be an integer as denoted in the model above or equivalently a real number in $[0, 1]$ denoting the cancellation rate. The model remains unchanged in complexity and can be used in either form since the accepted orders (denominator of cancel rate) can be normalized out everywhere. Since these are stochastic optimization problems in reality and there can be no guarantees on future variations, it is acceptable if the cancellations or cancellation rate are a few percentage points off from the estimated target C when it is in production; thus, this is viewed as a guideline target as opposed to a hard constraint. The cancellation rate used by retailers are typically in the range of 2%-15% of SFS demand which might translate to .2%-1.5% of all online demand depending on SFS penetration, whether the retailer is a high-end or discount store and practical operational efficiencies.

[1]For the sake of simplicity we are ignoring quantity ordered whereas in reality, computing an order status and quantity can be more complicated.

[2]In reality, there are multiple cancellation codes. We filter out irrelevant cancels from our analysis.

2.4 Optimization Models

In this section we address tractability of the underlying optimization problem and a pure optimization approach for setting thresholds.

2.4.1 Tractability

Theoretical Tractability

To formulate the threshold choice model as an integer program, we assume an upper bound U on the thresholds we can set. This gives us to the following formulation, where $x_{i,j}$ represents the choice that the threshold for item i is set to inventory value j. The data coefficients $r_{i,j}$ and $c_{i,j}$ correspond to the revenue and cancel rate for item i if the threshold is set to j, and are estimated from data.

$$\begin{aligned} \max \sum_{i=1}^{I}\sum_{j=1}^{U} r_{i,j}x_{i,j} \\ \text{such that} \sum_{i=1}^{I}\sum_{j=1}^{U} c_{i,j}x_{i,j} \le C \\ \sum_{j=1}^{U} x_{i,j} = 1 \ \forall i \in [I] \\ x_{i,j} \in \{0,1\} \ \forall i \in [I],\ \forall j \in [U] \end{aligned} \tag{2.1}$$

We will at times refer to integer program 2.1 as the Knapsack Threshold Problem or Knapsack Threshold IP. Since the constraint $\sum_{i=1}^{I}\sum_{j=1}^{U} c_{i,j}x_{i,j} \le C$ represents a knapsack constraint, it is not difficult to derive the following theorem from known results [35].

Theorem 5. *The linear programming relaxation of the integer program 2.1 has an optimal solution with fractional variables corresponding to at most one item. Given an optimal solution with fractional variables corresponding to multiple items, there is a $O(U \cdot I)$ algorithm to convert this solution to one with fractional variables corresponding to at most one item.*

Proof. We prove this by demonstrating that if there exists an optimal solution with fractional variables corresponding to more than one item, we can convert this solution into one with fewer fractional variables. Let two of the items with fractional variables be items i and j, such that $x_{i,a}$, $x_{i,b}$, $x_{j,c}$, and $x_{j,d}$ are all in the range $(0, 1)$. Our intention is to perturb these fractional variables so at least one variable becomes 0 or 1, all constraints remain satisfied, and the objective does not decrease.

Consider a new LP solution y, where $y = x$ except $y_{i,a} = x_{i,a} + \epsilon_1$, $y_{i,b} = x_{i,b} - \epsilon_1$, $y_{j,c} = x_{j,c} + \epsilon_2$, $y_{j,d} = x_{j,d} - \epsilon_2$. We wish to find ϵ_1, ϵ_2 so that y satisfies the model's constraints. For notational convenience let $c_1 = c_{i,a} - c_{i,b}$ and $c_2 = c_{j,c} - c_{j,d}$. Then we need ϵ_1 and ϵ_2 to satisfy the equation $c_1\epsilon_1 + c_2\epsilon_2 = 0$.

This is a single linear equation with two free variables, so it is possible to define ϵ_2 in terms of ϵ_1 and consider all solutions to these equations as determined by ϵ_1. In other words, we can define $x'(\epsilon)$, as $x'(\epsilon) = x$ except $x'(\epsilon)_{i,a} = x_{i,a} + \epsilon$, $x'(\epsilon)_{i,b} = x_{i,b} - \epsilon$, $x'(\epsilon)_{j,c} = x_{j,c} + f(\epsilon)$, $x'(\epsilon)_{j,d} = x_{j,d} - f(\epsilon)$ where $f(\epsilon)$ ensures that equation $c_1\epsilon + c_2 f(\epsilon) = 0$ remains satisfied.

Let $\epsilon' = \max(\epsilon : \; x'(\epsilon) \in [0,1]^{U\cdot I}$and $x'(-\epsilon) \in [0,1]^{U\cdot I})$ Intuitively, ϵ' will be the largest value such that both $x'(-\epsilon')$ and $x'(\epsilon')$ will be feasible solutions to the LP relaxation of Equations 2.1. Consequently, one of $x'(-\epsilon')$ or $x'(\epsilon')$ will have fewer fractional variables than x. From the definition of $f(\epsilon)$ we see if we set ϵ to either ϵ' or $-\epsilon'$ that both knapsack constraints of the LP is still satisfied, and x' is feasible because $|\epsilon|$ is such that all variables remain in the range $[0,1]$, but at least one of the fractional variables is now exactly 0 or 1. If $\epsilon(r_{i,a} - r_{i,b}) + f(\epsilon)(r_{j,c} - r_{j,d}) = 0$, then setting ϵ to one of ϵ' or $-\epsilon'$ results in a feasible x' with fewer fractional variables than x. Otherwise, setting ϵ to one of ϵ' or $-\epsilon'$ results in a feasible x' with a higher objective function value than x, contradicting the optimality of x. To convert an optimal solution with too many fractional variables into the desired optimal solution we can repeat the process shown in this proof over all fractional variables except for those corresponding to the last fractional item. □

Corollary 6. *If the functions $c_i()$ and $r_i()$ that we linearize to form the integer programming formulation of the Knapsack Threshold Problem are convex and concave, respectively, then the linear programming relaxation of the Knapsack Threshold Problem has an optimal solution with at most two fractional variables.*

Proof. Consider the LP solution guaranteed by Theorem 5 with fractional variables corresponding to at most one item. If this solution has more than two fractional variables, consider three of them, $x_{i,a}$, $x_{i,b}$, and $x_{i,c}$ such that $0 < x_{i,\alpha} + x_{i,\beta} + x_{i,\gamma} \leq 1$. Assume without loss of generality that $\alpha \leq \beta \leq \gamma$ and so by the convexity and concavity of $c_i()$ and $r_i()$ we know that $\lambda c_{i,\alpha} + (1-\lambda)c_{i,\gamma} = \lambda c_i(\alpha) + (1-\lambda)c_i(\gamma) \geq c_i(\lambda\alpha + (1-\lambda)\gamma)$ and $\lambda r_{i,\alpha} + (1-\lambda)r_{i,\gamma} = \lambda r_i(\alpha) + (1-\lambda)r_i(\gamma) \leq r_i(\lambda\alpha + (1-\lambda)\gamma)$. Let $\lambda = \frac{\beta-\gamma}{\alpha-\gamma}$ so that $\lambda\alpha + (1-\lambda)\gamma = \beta$. Then $\lambda c_{i,\alpha} + (1-\lambda)c_{i,\gamma} \geq c_i(\beta) = c_{i,\beta}$ and $\lambda r_{i,\alpha} + (1-\lambda)r_{i,\gamma} \leq r_i(\beta) = r_{i,\beta}$. Then, consider a policy $x'(\epsilon)$ where $x'(\epsilon) = x$ except $x'(\epsilon)_{i,\alpha} = x_{i,\alpha} - \lambda\epsilon$, $x'(\epsilon)_{i,\beta} = x_{i,\beta} + \epsilon$, and $x'(\epsilon)_{i,\gamma} = x_{i,\gamma} - (1-\lambda)\epsilon$ and $\epsilon > 0$. The constraint $\sum_{j=1}^{u} x_{i,j} = 1$ is still satisfied by $x'(\epsilon)$. The knapsack constraint is still satisfied because replacing x with $x'(\epsilon)$ decreases the left-hand side of the constraint:

$$\begin{aligned}
& (x_{i,\alpha} - \lambda\epsilon)c_{i,\alpha} + (x_{i,\beta} + \epsilon)c_{i,\beta} + (x_{i,\gamma} - (1-\lambda)\epsilon)c_{i,\gamma} \\
= \; & x_{i,\alpha}c_{i,\alpha} + x_{i,\beta}c_{i,\beta} + x_{i,\gamma}c_{i,\gamma} + \epsilon(c_{i,\beta} - (\lambda c_{i,\alpha} + (1-\lambda)c_{i,\gamma})) \\
\leq \; & x_{i,\alpha}c_{i,\alpha} + x_{i,\beta}c_{i,\beta} + x_{i,\gamma}c_{i,\gamma} + \epsilon(c_{i,\beta} - c_i(\lambda\alpha + (1-\lambda)\gamma)) \\
= \; & x_{i,\alpha}c_{i,\alpha} + x_{i,\beta}c_{i,\beta} + x_{i,\gamma}c_{i,\gamma} + \epsilon(c_{i,\beta} - c_{i,\beta}) \\
= \; & x_{i,\alpha}c_{i,\alpha} + x_{i,\beta}c_{i,\beta} + x_{i,\gamma}c_{i,\gamma}.
\end{aligned}$$

Let $\epsilon = \min\{\frac{x_{i,\alpha}}{\lambda}, 1 - x_{i,\beta}, \frac{x_{i,\gamma}}{1-\lambda}\}$. For this selection of ϵ, the constraints $x_{i,j} \in \{0,1\}$ for $j \in \{\alpha, \beta, \gamma\}$also continue to be satisfied, so $x'(\epsilon)$ is a feasible solution. Let $OPT = \sum_{i=1}^{n} \sum_{j=1}^{u} r_{i,j} x_{i,j}$, the value of the objective function under solution x. Since x is assumed to be optimal, we see that

$$\begin{aligned} OPT \geq & OPT - (x_{i,\alpha} r_{i,\alpha} + x_{i,\beta} r_{i,\beta} + x_{i,\gamma} r_{i,\gamma}) + \\ & ((x_{i,\alpha} - \lambda\epsilon) r_{i,\alpha} + (x_{i,\beta} + \epsilon) r_{i,\beta} + (x_{i,\gamma} - (1-\lambda)\epsilon) r_{i,\gamma}) \end{aligned}$$

which is equivalent to the following by rearranging terms:

$$\begin{aligned} x_{i,\alpha} r_{i,\alpha} + x_{i,\beta} r_{i,\beta} + x_{i,\gamma} r_{i,\gamma} \geq & (x_{i,\alpha} - \lambda\epsilon) r_{i,\alpha} + (x_{i,\beta} + \epsilon) r_{i,\beta} + (x_{i,\gamma} - (1-\lambda)\epsilon) r_{i,\gamma} \\ 0 \geq & \epsilon (r_{i,\beta} - (\lambda r_{i,\alpha} + (1-\lambda) r_{i,\gamma})) \\ \lambda r_{i,\alpha} + (1-\lambda) r_{i,\gamma} \geq & r_{i,\beta}. \end{aligned}$$

Earlier we observed that $\lambda r_{i,\alpha} + (1-\lambda) r_{i,\gamma} \leq r_{i,\beta}$ from the concavity of $r_i()$, so it must be the case that $\lambda r_{i,\alpha} + (1-\lambda) r_{i,\gamma} = r_{i,\beta}$. Then $x'(\epsilon)$ is also an optimal solution with fewer fractional variables than x. □

Corollary 7. *If the functions $c_i()$ and $r_i()$ that we linearize to form the integer programming formulation of the Knapsack Threshold Problem are convex and concave, respectively, then the linear programming relaxation of the Knapsack Threshold Problem has an optimal solution with at most two fractional variables. Furthermore, these two fractional variables correspond to neighboring thresholds for the same item.*

Proof. The first part of this corollary is given to us by Corollary 6. Consider the optimal LP solution x guaranteed to us by Corollary 6. Suppose that the fractional variables in this solution do not correspond to adjacent thresholds, then let $x_{i,\alpha}$ and $x_{i,\gamma}$ be fractional such that $x_{i,\beta} = 0$ and $\alpha < \beta < \gamma$. Let $\lambda = \frac{\beta-\gamma}{\alpha-\gamma}$ so that $\lambda\alpha + (1-\lambda)\gamma = \beta$. Then, consider a policy $x'(\epsilon)$ where $x'(\epsilon) = x$ except $x'(\epsilon)_{i,\alpha} = x_{i,\alpha} - \lambda\epsilon$, $x'(\epsilon)_{i,\beta} = x_{i,\beta} + \epsilon$, and $x'(\epsilon)_{i,\gamma} = x_{i,\gamma} - (1-\lambda)\epsilon$. If we set $\epsilon = \min\{\frac{x_{i,\alpha}}{\lambda}, 1, \frac{x_{i,\gamma}}{1-\lambda}\}$, then from the calculations in the proof of Corollary 6 we see that x' is feasible and also optimal. I claim that at least one of $x'(\epsilon)_{i,\alpha}$ or $x'(\epsilon)_{i,\gamma}$ will be 0. If $\epsilon = \frac{x_{i,\alpha}}{\lambda}$ then $x'(\epsilon)_{i,\alpha} = x_{i,\alpha} - \lambda\epsilon = x_{i,\alpha} - \lambda\frac{x_{i,\alpha}}{\lambda} = 0$. If $\epsilon = \frac{x_{i,\gamma}}{1-\lambda}$ then $x'(\epsilon)_{i,\gamma} = x_{i,\gamma} - (1-\lambda)\epsilon = x_{i,\gamma} - (1-\lambda)\frac{x_{i,\gamma}}{1-\lambda} = 0$. Lastly, if $\epsilon = 1$ then $x'(\epsilon)_{i,\beta} = 1$. Since $\sum_{j=1}^{u} x'(\epsilon)_{i,j} = 1$ then $x'(\epsilon)_{i,\alpha} = x'(\epsilon)_{i,\gamma} = 0$. Then, $x'(\epsilon)$ is an optimal solution with either fewer fractional variables than x or it continues to have two fractional variables corresponding to thresholds closer to each other than the thresholds with fractional variables in solution x. This process can be repeated until we have a solution with no fractional variables or two adjacent fractional variables, proving the corollary. □

Theorem 5 and its corollaries are of practical significance because they provides a high-level guarantee on the value of a policy dictated by the solution to the linear programming relaxation. In Chapter 3 we introduce SafetyNet, an end-to-end neural network model for this same setting of omni-channel fulfillment. These findings explain how the SafetyNet models effectively use the linear programming relaxation of the Knapsack Threshold Problem to find near-integer threshold policies. The optimum solution is of course the solution with integer variables. However, this theorem shows us that when I is large, say you have 100,000 items you can immediately find a solution with fractional variables corresponding to at most one item and the rest being integer.

The contribution of those fractional variables, for at most two items among the entire set of items carried by the retailer is practically tiny to the revenue or cancellations of the retailer. Moreover, heuristic methods like branch-and-bound typically solve the relaxation and find integer variables quite easily because the number of variables that need to be rounded is very small in each iteration. These theoretical results show that although technically the problem is NP-complete, it is tractable in practice.

Empirical Tractability

Given the above results, it is not surprising when we report that commercial integer programming solvers are able to handle relatively large instances of our knapsack threshold problem with ease. Running on a computer with a 2.50 GHz CPU and 16 GB RAM, Gurobi 6.5.0 was able to solve simulated problem instances where $I = 500,000$, $U = 10$ in under 15 seconds, and instances where $I = 1,000,000$, $U = 10$ in less than 1 minute. In reality, the number of products tracked in a typical large retailer is roughly about 2-5M across their network. However, the number of products that are actively sold at any time, with inventory greater than 0 is only about 200-500K. Out of these, some simple practical methods to merge very similar products to treat them as a single category reduces the item space by another order of magnitude which makes the above problem sizes fully realistic.

Before we proceed to describe methods for specifying parametric forms for the various coefficients $r_{i,j}$ and $c_{i,j}$ and estimating them in the next section, we first describe a non-parametric method that uses sample data (as an empirical distribution) to solve a pure optimization problem of choosing the thresholds for the items.

2.4.2 Pure Optimization Model: Sample Average Approximation

Sample Average Approximation (SAA) is a Monte Carlo simulation method that approximates an expected value function by taking a random sample of observations and computing the corresponding sample average. As an illustration, an iid sample of the order data can be used to generate a simple, unbiased estimate of $r_{i,j}$ by computing the revenue from all orders accepted for item i when its inventory is at least j in the sample. However, our goal is to compute the thresholds for the items, so we use these samples as trial data points for a large optimization model that chooses thresholds.

Suppose we are given the following data about a large stream of orders: $c_{i,j,k}$ is 1 if order (i, j, k), the kth observed order of item i at inventory level j, is cancelled and 0 otherwise. $r_{i,j,k} = (1 - c_{i,j,k})p_{i,j,k}$ where $p_{i,j,k}$ is the price of order (i, j, k). Now by setting $c_{i,j} = \sum_{t=1}^{j} \sum_{k=1}^{K_t} c_{i,t,k}$, and $r_{i,j} = \sum_{t=1}^{j} \sum_{k=1}^{K_t} r_{i,t,k}$ $\forall i, j$, we can use these estimated coefficients in solving the integer program 2.1.

As the size of the iid sample of demand approaches infinity, the estimates of coefficients $c_{i,j}$ and $r_{i,j}$ converge to the true expected values of cancels and revenue respectively, for item i when its threshold is j, due to the Law of Large Numbers. In turn, the SAA method using these coefficients can be shown to converge to optimal threshold values. There is an extensive literature on SAA methods that provide stronger convergence guarantees under more restrictive assumptions on our data (See [39] or [36]).

2.5 Supervised Learning

We considered several supervised learning techniques to identify orders likely to be canceled. When using supervised learning techniques to predict cancellations, an important set of features is the product ontology. Each product is classified at a number of different levels of specificity which form a tree structure. The lower level features of the product ontology strictly determine the higher level features in the ontology, meaning that for many classification methods at most one ontology feature can be included in the learning model without violating assumptions about feature independence. To protect the anonymity of the retailer providing us data, we will call the features in the product ontology A, B, C, and D, where A is the highest level feature in the ontology, and D is the lowest level.

In our Separate Estimation and Optimization methods presented in Section 2.6, we rely on logistic regressions for each item category (determined by a single ontology feature) to estimate cancel rates as a function of inventory level. This model fits the real retail data nicely (see Figure 2.4) and also ensures that cancel rate as a function of inventory is strictly decreasing within each category. This is an intuitively desirable property and also one that becomes mathematically required for the methods we consider later on in Chapter 3 to work correctly. In the rest of this section we present some findings from our exploratory work in estimating cancel rates. Most significant to the rest of the work presented in this chapter, we identified only a very modest benefit from using cancellation prediction models that consider multiple ontology features.

2.5.1 Comparison of Cancel Rate Prediction Methods

As a baseline we trained multiple naive Bayes classifiers, each using a different ontology feature along with a set of other predictive features (inventory count, price

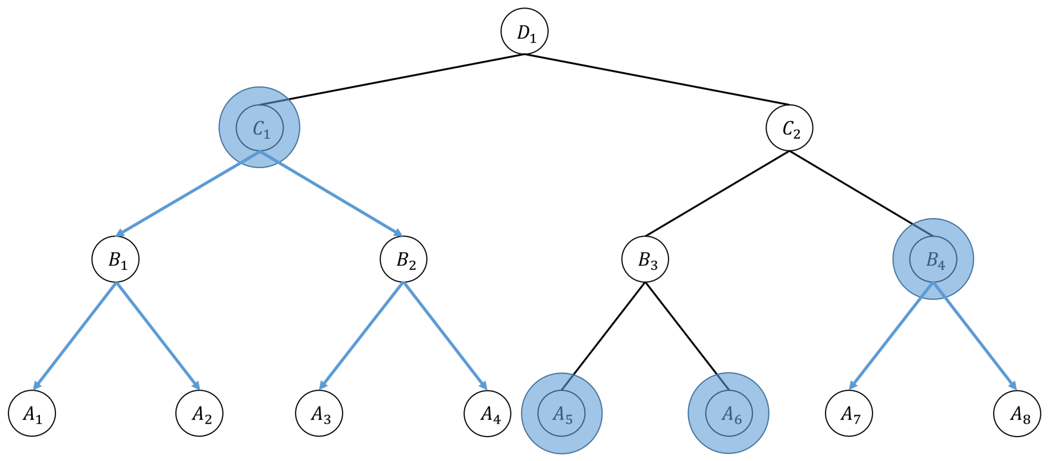


FIGURE 2.1: Example of dynamic programming method for supervised learning. Blue circles denote the nodes in set S for this instance.

status, vendor ID, order quantity). Our metric for performance of these models is the area under the curve (AUC) of its receiver operating characteristic (ROC) curve. Without any modifications, several of these baseline models performed well at identifying cancellations. The best of these models has an AUC of .7891, computed using 10-fold cross-validation. In the remainder of this section, we present new techniques which we hope can lead to improvements over this baseline (see Figure 2.2 for their relative performances).

The first alternative method we consider is a dynamic programming-based best-of-many approach. Our goal is to choose the best of our baseline classifiers to use on this order. We take the tree defined by the features of the ontology and give each node an error value, which is the number of mistakes made by the baseline classifier that uses the ontology feature at the same level in the tree as the current node on the orders that match the value of the current node. Our objective is to find a minimum cost set S of nodes such that every leaf node is either in S or has an ancestor that is in S. For each order, we find the node in set S whose feature matches with the order and we use the predictive model corresponding to this feature to predict the cancellation probability of the order. Using standard dynamic programming techniques it is straightforward to compute S. Figure 2.1 visualizes a possible set S for an example of a product ontology (real product ontologies are not necessarily binary trees like this simple example). Unfortunately, this does not seem to provide improvement over the baseline models. This best-of-many approach produced a ROC curve with an AUC of .7839, also using 10-fold cross-validation.

The next method takes a convex combination of the predictions output by the different baseline models to predict the outcome of an order. In principle, this approach allows different products in the ontology to use different weights in the convex combination. However, at present, we only have implemented a simple average

Model	AUC
Ontology Feature A	.7860
Ontology Feature B	.7891
Ontology Feature C	.7840
Ontology Feature D	.7810
Dynamic Programming	.7832
Shrinkage	.7897

FIGURE 2.2: Summary of supervised learning results

over several of higher-performing baseline models. Still, this is able to give a modest improvement over the best baseline model, with an AUC of .7897 using 10-fold cross-validation.

In our implementations of algorithms to produce threshold policies we have decided to use models of cancel rates that consider a single ontology feature. We have also chosen to use logistic regression models of the cancel rate to ensure the cancel rate functions fed into our optimization are strictly decreasing as inventory increases. The findings in this section may still provide insight into predicting cancellations in ship-from-store online retail programs that may be incorporated into future threshold optimization algorithms.

2.6 Separate Estimation and Optimization

In this section, we describe a class of models to specify and learn the coefficients $c_{i,j}$ and $r_{i,j}$. While there are several possible ways of performing this, we show two different approaches in this section – Maximum Likelihood (MLE) and Expectation-Maximization (EM) – motivated by theory and practice that might be of interest. In the next chapter, we introduce the SafetyNet class of models that perform both the parametric estimation and the optimization of the thresholds described in Section 2.4 together in one end-to-end pass using neural networks. We then compare the relative accuracy of these various methods, the pure optimization SAA method from the previous section, the two estimation methods MLE and EM from this section feeding the IP optimization, and the end-to-end method from the next section in Section 3.6.

2.6.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) can be used to generate more accurate estimates of our coefficients when we have a parametric model of the distributions that determine our coefficients. To use this method, we first introduce the estimation of a demand variable that will be useful later in estimating the other coefficients.

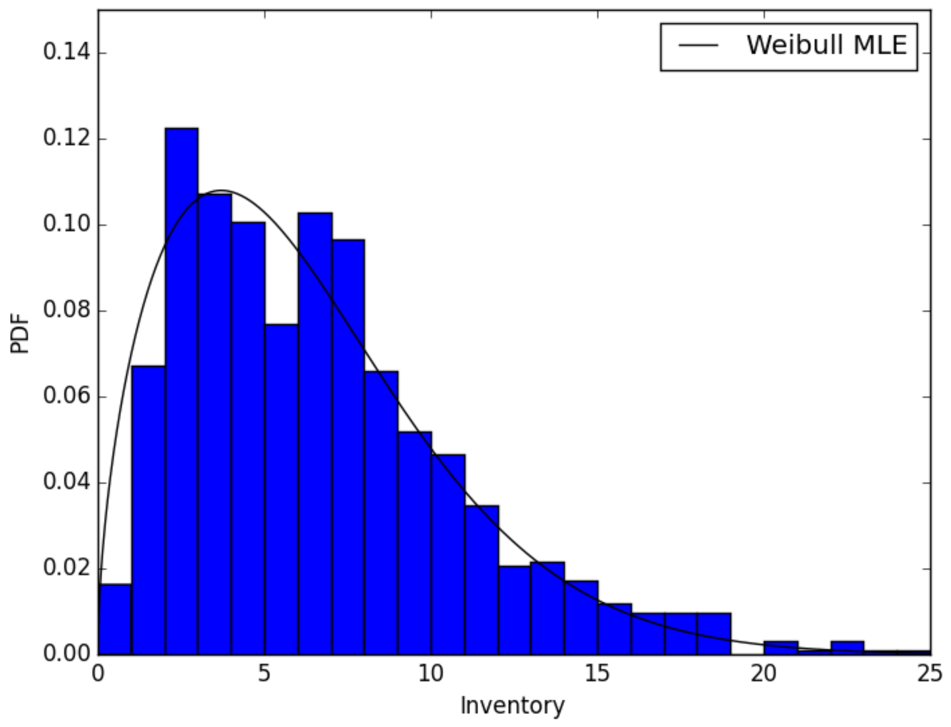


FIGURE 2.3: Example fit of Weibull distribution of inventory counts to real Onera data

Clustering and Demand Estimation

To help with demand estimation, we collapse the set of items I into categories or clusters. In practice this is performed using the product feed features to generate clusters of similar products which are then assumed to be drawn iid from the cluster's demand distribution. Henceforth, we will use clusters or items I interchangeably. Demand for an item within each cluster is modeled by a Weibull distribution. The Weibull distribution is appropriate as it is a flexible class of distributions capable of modeling both symmetric and right skewed distributions. Figure 2.3 is a histogram of inventory counts for a representative category of items at one of Onera's clients along with a fitted Weibull distribution, motivating this methodology.

We use MLE to obtain the parameters λ_i (scale) and k_i (shape) of the Weibull distribution for each cluster. We are using a continuous approximation of a discrete distribution with the Weibull distribution, so to estimate $d_{i,j}$, the expected demand for cluster i at inventory level j, we multiply $P_i(j-1 \leq x \leq j) = e^{-(j/\lambda_i)^{k_i}} - e^{-((j-1)/\lambda_i)^{k_i}}$ by the total volume of orders for item i in our training sample.

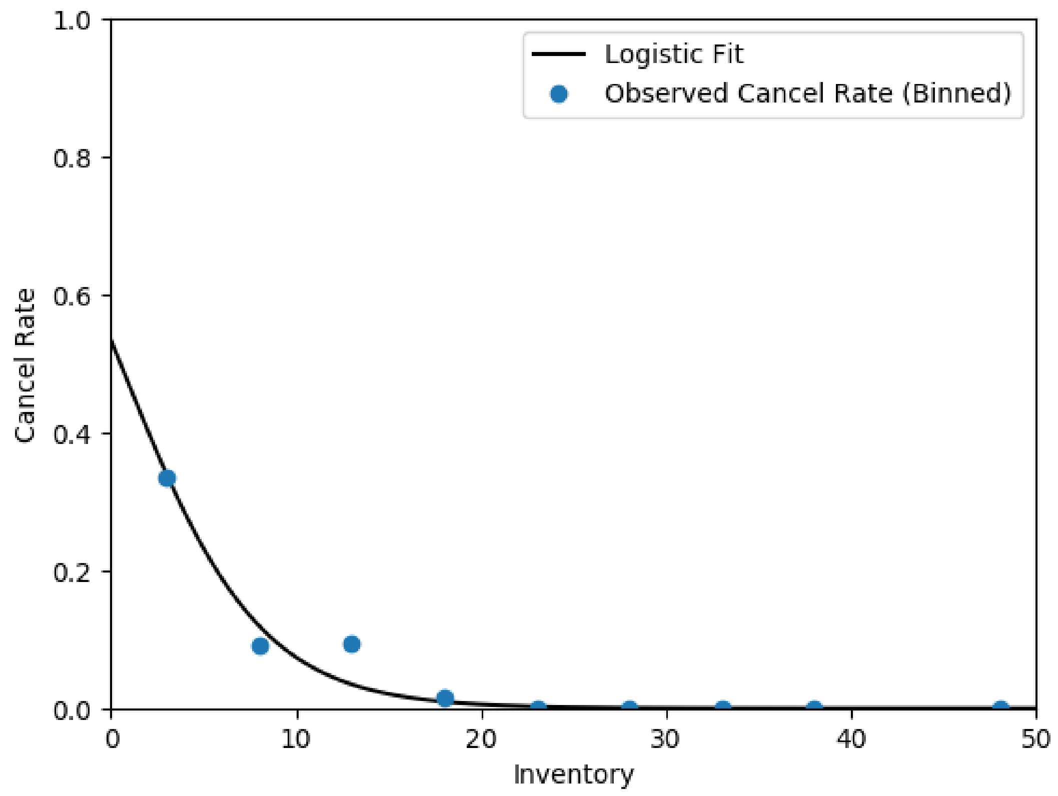


FIGURE 2.4: Example fit of logistic function to observed cancel rates on real Onera data

Cancel Rate Estimation Details

We model cancel rates as a function of inventory x for an item in cluster i by the logistic function $f_i(x) = \frac{1}{1+e^{-(\beta_{0i}+\beta_{1i}x)}}$. In real-life omni-channel SFS programs, cancelations are often due to phantom inventory, where inventory is listed in a database as available but is not actually available. Consequently, orders accepted for items with lower listed inventory levels are more likely to be cancelled, motivating our use of a logistic function to model cancel rates. We use the machine learning library Scikit-learn [46] to estimate parameters β_{0i} and β_{1i} for each cluster i by MLE. We use this function to compute our cancellation coefficients: $c_{i,j} = \sum_{x=j}^{U} f_i(x)d_{i,x}$ where $d_{i,x}$ is the demand estimate described above. Figure 2.4 compares the fit of a Logistic function to the observed cancel rates for a category of items at one of Onera's clients.

2.6.2 Expectation-Maximization

As mentioned earlier, historical data is intrinsically truncated because orders are not observed for items i with inventory lower than the thresholds t_i that were set historically. So far, our methods do not fully account for this truncation in the demand at lower inventory counts. We use Expectation-Maximization (EM) to perform MLE on a distribution that has been truncated at a known point. MLE SafetyNet, presented

```
function EM(Data, Thresh)
    $\lambda, k \leftarrow$ MLE(Data)
    $\lambda_{\text{prev}}, k_{\text{prev}} \leftarrow \infty, \infty$
    while $\max(|\lambda - \lambda_{\text{prev}}|, |k - k_{\text{prev}}|) > \epsilon$ do
        $\lambda_{\text{prev}}, k_{\text{prev}} \leftarrow \lambda, k$
        EstTruncData $\leftarrow$ MaximizationStep(Data, $\lambda, k$, Thresh)
        $\lambda, k \leftarrow$ ExpectationStep(Data, EstTruncData)
    end while
    return $\lambda, k$
end function

function MaximizationStep(Data, $\lambda, k$, Thresh)
    PctTrunc $\leftarrow$ CDF(Thresh, $\lambda, k$)
    Out $\leftarrow \{\}$
    for Inv $= 1$ to Thresh do
        Expect $\leftarrow$ (CDF(Inv, $\lambda, k$) $-$ CDF(Inv $- 1, \lambda, k$)) $\cdot \frac{|\text{Data}|}{1-\text{PctTrunc}}$
        Out $\leftarrow$ Out + Inv$^{|\text{Expect}|}$
        (Append expected value of orders at
        inventory level Inv to Out data)
    end for
    return Out
end function

function ExpectationStep(Data, EstTruncData)
    EstFullData $\leftarrow$ Concat(Data, EstTruncData)
    return MLE(EstFullData)
end function
```

FIGURE 2.5: EM Pseudocode

in Section 3.5, can also be used to estimate distributions from data with multiple points of truncation in a conceptually similar manner.

EM methods converge to an MLE estimate of the un-truncated distribution under general conditions that include this setting [10]. In our experiments we iterate through the Maximization and Expectation steps until the Weibull Distribution parameters λ_i (scale) and k_i (shape) converge and stop changing significantly between iterations. We use EM specifically to get an MLE estimate of a Weibull distribution of inventory within clusters of items where the order data for the clusters are truncated by previous threshold values. We outline the conceptual algorithm for the two steps below. Psuedocode for this method is shown in Figure 2.5. More detail on the EM method we implement in our empirical studies can be found in prior work [10].

Maximization Step

The Maximization step for this use case is largely the same as what is described in Section 2.6.1. The only significant change is that the sample data we use to fit our

Weibull distributions includes both the original data and the additional data points that are added during the Expectation step.

Expectation Step

The Expectation Step considers each inventory level below the cluster's maximum threshold value throughout the training sample, the point where our data gets truncated. At each of these inventory levels we compute the expected level of demand based on the Weibull distribution fit during the most recent Maximization step. If this is greater than the observed demand we add these values as additional data points in our training sample to be used in the following Maximization step.

2.6.3 Proxy Maximum Likelihood Estimation

The Expectation-Maximization estimation method from the previous section aims to correct for the effect of truncation in the orders data. In this section we present the Proxy-MLE method of parameter estimation, which accounts for data truncation by estimating the distribution of inventory levels and mean demand rates using inventory data as a proxy for orders data. This method has the advantage that inventory data is independent of the retailer's threshold policy. However, a potential downside to this method is that the estimates mades from inventory data may not exactly correspond to the ground truth of the orders data.

We use the same parametric model as described in the MLE estimation method. A Weibull distribution over the inventory level of items in each cluster is estimated by MLE, though for this method we use the inventory data as the source of data samples. We use up to three months of inventory data collected prior to the time of estimation, and we filter out UPCs for which no sales have been recorded in the previous five months.

Overall cluster demand rates are estimated at the UPC level and then aggregated at the cluster level. We use the inventory data to identify the dates when each UPC's inventory count was above its threshold value, meaning it had been available for purchase online. We take the mean of the volume of orders recorded across the days when inventory was at or above the relevant cluster's threshold value as the mean demand rate for each UPC. These demand rate estimates are exactly the MLE estimates of Poisson rate parameters if demand was modeled as an independent Poisson process for each UPC.

2.7 Dynamic Threshold Policy

The previous techniques we consider all rely on using historical data to determine a policy for the present. However, if many orders begin to cancel for reasons that are not reflected in changes to the input features of these models, we may not be able to adapt to these changes quickly enough. The adaptability and simplicity of this dynamic policy is its primary appeal; we don't expect this type of policy to outperform the other methods if cancellations are following patterns found in historical data.

We observe in our data that the probability of fulfillment for an order as a function of its threshold can be closely approximated by a logistic function. We can use this observation to define a formal model of this policy for theoretical analysis. Let our logistic function be $p(x) = \frac{1}{1+e^{-\frac{x}{b}}}$, where x is the current value of the threshold and b is a constant.

Definition 6. *A $(+a, -b)$ policy is a dynamic threshold policy, where the threshold is increased by a each time an order is canceled and decreased by b each time an order is successfully filled.*

We then observe that if we want our long-term cancellation rate to be λ, a $(+\epsilon, -\frac{\lambda}{1-\lambda}\epsilon)$ policy has many desirable properties.

Lemma 6. *Let $x = p^{-1}(1-\lambda)$ and let x' be the threshold after one iteration of the $(+\epsilon, -\frac{\lambda}{1-\lambda}\epsilon)$ policy. Then $E[x'] = x$.*

Proof. The proof follows immediately from the value of x:

$$\begin{aligned}
E[x'] =& x + \epsilon(1-p(x)) - \frac{\lambda}{1-\lambda}\epsilon p(x) \\
=& x + \epsilon\lambda - \frac{\lambda}{1-\lambda}\epsilon(1-\lambda) \\
=& x.
\end{aligned}$$

$\square$

Lemma 7. *Suppose $x > p^{-1}(1-\lambda)$ and let x' be the threshold after one iteration of the $(+\epsilon, -\frac{\lambda}{1-\lambda}\epsilon)$ policy. Then $E[x'] < x$.*

Proof. The proof is similar to that of Lemma 2:

$$\begin{aligned}
E[x'] =& x + \epsilon(1-p(x)) - \frac{\lambda}{1-\lambda}\epsilon p(x) \\
=& x + \epsilon - \epsilon p(x) - \frac{\lambda}{1-\lambda}\epsilon p(x) \\
=& x + \epsilon - \frac{1-\lambda}{1-\lambda}\epsilon p(x) - \frac{\lambda}{1-\lambda}\epsilon p(x) \\
=& x + \epsilon - \frac{\epsilon p(x)}{1-\lambda} \\
<& x + \epsilon - \frac{\epsilon(1-\lambda)}{1-\lambda} \\
=& x.
\end{aligned}$$

$\square$

Lemma 8. *Suppose $x < p^{-1}(1-\lambda)$ and let x' be the threshold after one iteration of the $(+\epsilon, -\frac{\lambda}{1-\lambda}\epsilon)$ policy. Then $E[x'] > x$.*

Proof. The argument is nearly identical to the proof of Lemma 3. □

However, if the retailer was aware of the function $p(x)$, they would be able to simply set the threshold to $p^{-1}(1-\lambda)$ and would not have to bother with a dynamic policy. Then, an interesting variant of this model is one in which the parameter b varies rather than remaining constant. In this case it is clear that a dynamic policy would be more effective.

2.8 Empirical Results

2.8.1 Methods

We use real Onera data to evaluate how effective our methods are at optimizing revenue while maintaining a pre-specified cancel and acceptance rate in realistic conditions. We compare the performance of our five methods – Onera-SAA, Onera-MLE, Onera-EM, Onera-Proxy, and Onera-Dynamic – for estimating the coefficients of this model to each other and to Retail-1-threshold policies which accept or reject orders for all items based on a single unified threshold. These Onera policies are implementations of each of the three statistical estimation methods described in Sections 2.6 and 2.7 : Sample Average Approximation, Maximum Likelihood Estimation, Expectation-Maximization, Proxy Maximum Likelihood Estimation, and Dynamic Threshold Policy, respectively.

The first two methods, Onera-SAA and Onera-MLE make no adjustments for data truncation and serve as benchmarks for what can be expected from optimization methods that do not model the effects of truncated demand. The Onera-EM method explicitly accounts for truncated demand, allowing us to assess the practical impact of explicitly accounting for truncation in our models. The Onera-Proxy method gets around the problem of data truncation by using non-truncated inventory data as a proxy for some of the information typically gathered from truncated orders data, and so this method's performance is a valuable point of comparison for the Onera-EM method. Lastly, the Onera-Dynamic method does not attempt to optimize for revenue in any way, but it is able dynamically adjust thresholds to track the target cancel rate, even if the data generation process changes over time.

A limitation of using real retail data is that our choice of thresholds cannot always influence which orders get truncated. However, using real data is important to demonstrate that our models and estimation methods can be applied in production systems and not just in a stylized environment. We are able to partially compensate for the truncated orders that are not observed in our data by artificially truncating orders that were below an artificial threshold we impose for our experiments. This artificial truncation creates an environment where we can at least partially observe the ground truth data that are not visible to our estimation algorithms.

We use a high-end multi-billion dollar annual online revenue fashion retailer to demonstrate our results. We used the SFS orders from 3 consecutive months of 2016

Simulation Results			
Policy	Cancels	Revenue	Increase
Retail-1-Threshold	6.19%	54.24%	N/A
Onera-SAA	7.09%	71.42%	31.65%
Onera-MLE	7.25%	75.75%	39.64%
Onera-EM	6.67%	72.29%	33.26%
Onera-Proxy	8.08%	78.52%	44.74%
Onera-Dynamic	6.35%	61.42%	13.22%

FIGURE 2.6: Simulation results for experiment with unadjusted cancel rate target

of this retailer as the data set for the analysis. We evaluate our methods one week at a time, allowing our models to train on all data prior to the evaluation week. In addition to naturally occurring data truncation, we exclude all orders whose listed inventory is less than 8 from the training data to insert an additional source of data truncation that is observable. This process truncates approximately 35% of the original training data. Across our 12 evaluation weeks we compare our Onera policies to a benchmark policy, Retail-1-Threshold, which sets a constant threshold of 10 to all orders.

We use the realized cancel rate of this Retail policy as inputs to our model and demonstrate that our methods can produce dominant policies. This target cancel rate is calculated from the artificially truncated training data collected prior to the first evaluation week. We note that this target cancel rate was 5.2%, which is somewhat lower than the cancel rate realized by this same policy across the weeks evaluated in the simulation, 6.2%. This suggests that the real-world circumstances driving the data generation process may be changing over time.

Additionally, we ran a second set of evaluations where we scale the cancel rate target of our Knapsack Threshold IP down by 30%, producing policies that are more conservative about allowing cancellations. This is to ensure that we produce a set of policies that adhere to the cancel rate limit and will strictly dominate the benchmark policy by achieving lower cancel rates as well as greater revenues.

2.8.2 Results

The Figures 2.6 and 2.7 show, for all policies tested, the mean cancel rates, mean revenues (as a percent of the maximum possible revenue), and percentage increase in revenue over the benchmark Retail-1-Threshold policy.

We find that our methods consistently produce sizable increases in revenue over the benchmark Retail-1-Threshold policy, with a minimum average revenue increase across all policies of 13.22%. However, especially when using unadjusted cancel rate targets, these policies tend to overshoot the stated cancel rate target. As a consequence, it it difficult to perform a direct comparison between the Onera policies and

Simulation Results - Adjusted Cancel Target			
Policy	Cancels	Revenue	Increase
Retail-1-Threshold	6.19%	54.24%	N/A
Onera-SAA	6.33%	66.28%	22.19%
Onera-MLE	5.97%	67.32%	24.10%
Onera-EM	5.70%	64.95%	19.73%
Onera-Proxy	6.07%	68.01%	25.36%
Onera-Dynamic	5.42%	54.83%	1.07%

FIGURE 2.7: Simulation results for experiment with cancel rate target shifted down 30%

the benchmark Retail-1-Threshold policy. We do see that the Onera-EM method, which attempts to correct for data truncation, has the closest cancel rate among Separate Estimation and Optimization (SEO) methods to that of the Retail-1-Threshold policy, but a smaller difference in cancel rates would be better. The Onera-Dynamic method performs relatively better than the other methods at tracking its target cancel rate (5.2%), as we might expect, but it does not appear to produce policies that approach Pareto efficiency. We see that the policies produced by Onera-EM with an adjusted cancel target dominate the Onera-Dynamic method by achieving a higher revenue percentage and a lower cancel rate.

When we scale down the cancel rate target in the Onera models by 30% we get policies that much more closely track the cancel rate of the Retail-1-Threshold policy. In particular, the Onera-EM method strictly dominates the Retail-1-Threshold policy, with a just slightly lower cancel rate and a 19.73% increase in revenue. In Onera's production systems they perform significant manual tuning to ensure their policies closely track the target cancel rate. Our results from the set of policies with adjusted cancel rates provide evidence that these models can be successfully used to substantially outperform the benchmark if the cancel rate target is tuned properly. For a typical retailer whose revenue is a billion dollars online, SFS can contribute to about 10-20% of the overall volume yielding about 100-200M in annual revenue. A 25% lift on a 200M annual revenue is a substantial 50M dollars in incremental revenue or 5% increase in the entire online revenue for the retailer.

2.9 Discussion

In this chapter, we introduced the problem of optimizing inventory for omni-channel fulfillment and presented knapsack-inspired threshold models as tools to solve this problem. We show theoretical and empirical evidence that these models can be formulated and solved efficiently. To account for demand data truncation that occurs in real-life settings, we provide two method to estimate the parameters that explicitly corrects for this factor. Finally, we use real retail data to show the significant lift in revenue that can be realized by using an intelligent model to optimize inventory thresholds for omni-channel fulfillment.

The results from our experiments also expose some limitations of these methods, which we will go on to address in Chapter 3. In particular, we see that running our algorithms without manual supervision can lead to threshold policies that are Pareto efficient – or at least close – but may not always track the cancel rate target specified by the retailer. Depending on the application, this may not be acceptable. An ideal algorithm would be able to measure how closely it tracks the target cancel rate and adjust its policies accordingly, rather than relying on manual tuning. This observation motivates the SafetyNet models that are the focus of Chapter 3. These SafetyNet models provide end-to-end learning methods that address these concerns.

Acknowledgments: Prof. R. Ravi and Dr. Srinath Sridhar are co-authors on this work. The real retail data analyzed in this chapter was graciously provided by Dr. Sridhar and his company Onera.

Chapter 3

Joint Estimation and Optimization of Knapsack Threshold Models

3.1 Introduction

Chapter 2 presents the problem of setting threshold policies to efficiently balance revenue and cancelled orders in an omni-channel retailing setting. We provided motivation for studying this problem, most notably that analytics consultancy Onera provides services related to this problem as one of its core products, and we introduced and evaluated several methods to algorithmically generate threshold policies. The work presented in this chapter is motivated by the limitations of the methods investigated in Chapter 2. These methods perform optimization of threshold policies and the underlying estimates of model parameters as separate steps in an algorithmic pipeline. A negative consequence from using this class of methods is small errors or biases in the estimation process can have unexpected effects on the final policy when these estimates are used to formulate the optimization problem responsible for producing this policy. We observed this, where some of our methods in Chapter 2 produced efficient policies, though manual tuning was required to find policies that closely tracked the pre-specified cancel rate targets.

In this chapter, we use artificial neural networks to construct end-to-end models that perform parameter estimation and threshold policy optimization in a single parametric model. We present GreedyNet, a neural network layer which performs differentiable optimization using a greedy algorithm. GreedyNet allows us to embed the argmax function of many optimization problems that can be solved exactly or approximately by a greedy algorithm, such as the threshold optimization problem introduced in Chapter 2, inside a neural network. We demonstrate the utility of GreedyNet as a component in SafetyNet, a network architecture that performs end-to-end optimization of threshold policies for the omni-channel fulfillment problem studied in Chapter 2.

3.1.1 Contributions

We make the following contributions in this chapter:

1. We introduce GreedyNet, a neural network layer for differentiable greedy optimization, and we provide technical details on its implementation.

2. Leveraging GreedyNet, we formulate network architecture SafetyNet, an end-to-end model for the omni-channel fulfillment problem. We provide technical details of several versions of the SafetyNet architecture and discuss their relative trade-offs.

3. Three versions of SafetyNet are evaluated on the same conditions used to test the separate estimation and optimization methods from Chapter 2. These evaluations involve tests using both real and simulated retail data, and provide empirical evidence that SafetyNet models find high-quality threshold policies which more closely track the target cancel rates than our earlier methods and benchmarks.

4. We compare the running time of GreedyNet with that of OptNet [5], an existing architecture for differentiable optimization. We present empirical results comparing the running time of SafetyNet models powered by both GreedyNet and OptNet. Our experiments show a 4x speedup in favor of GreedyNet on moderate-sized instances, and greater gains in performance as the instance size increases.

3.2 Background

3.2.1 Neural Networks

Our work on GreedyNet and SafetyNet builds on a large body of research on artificial neural networks. A good source for a more general overview of neural networks is the Deep Learning textbook by Goodfellow et al. [24]. Within the context of this chapter, it is most useful to think of neural networks as a flexible class of parametric functions. Each neural network can be represented as computational graph. In this representation, the neural network is a directed acyclic graph (DAG), where each edge represents a vector and each node in the graph represents a differentiable function which receives a set of vectors as input and returns a vector as output. There are more general representations of neural networks in which edges of the computational graph pass multi-dimensional objects known as tensors between nodes [24, Chapter 2], but it is sufficient to restrict ourselves to vectors for the purposes of this background section. The DAG structure of this graph allows us to sort the graph topologically and pass information through this graph in both directions.

The typical training process for fitting a neural network is to minimize a loss function using a first-order optimization method such as Stochastic Gradient Descent [9]. There are also many newer optimization methods such as Adam [38], Nesterov Momentum [52], and RMSProp [53] that have become very popular for training neural networks in recent years. The gradient estimates required to implement these optimization methods are computed by the neural network using operations referred to as forward and backward passes. The forward pass operation evaluates the current function represented by the neural network for a set of input values and computes a scalar-valued loss function from this output. Intuitively, the backward pass sends information backwards through the graph and determines how much influence each parameter had on the loss function for the input values evaluated during the last forward pass. More precisely, the backward pass recursively applies the chain rule of calculus backwards through the network to compute the partial derivative of the loss function with respect to each parameter of the network. Upon completion of a backward pass, the vector of these partial derivatives forms the gradient estimate needed to run an iteration of Stochastic Gradient Descent or some other first-order optimization method. This process is repeated until the parameters of the network converge such that the loss function is at a local minimum.

A forward pass evaluates the neural network function for a set of input values by computing the composition of all the functions contained inside the graph, as ordered by the topological ordering. Nodes in the DAG may contain additional stored parameters that are inputs to that node's differentiable function. The nodes of the DAG with no incoming edges initiate the forward pass operation by passing input values or stored parameters to the next layer of the model. The nodes of the DAG with no outgoing edges return the output value from the neural network function. We can specify that there will be only a single node with no outgoing edges and it will return a scalar loss or objective function. Alternative formulations of neural networks allow for one or many multi-dimensional output nodes, and a scalar loss or objective value can be computed with a function that exists outside of the neural network.

The other primary requirement of a standard neural network is that each node must contain a differentiable function so that a backward pass may flow backwards through the network. Satisfying this requirement allows us to recursively apply the chain rule backwards through the network DAG to compute gradients of the network output function with respect to the network's parameters. This process is known as backpropogation in the machine learning community, and we will spend the next several paragraphs describing this process in more detail.

Concretely, consider a neural network with m layers in the topological ordering of its DAG, where each layer $i \in [1, \dots, m]$ has n_i intermediate vectors, $x_{i,1}, \dots, x_{i,n_i}$ as inputs. These intermediate vectors are the vectors represented by the edges of the DAG and get computed during each forward pass through the network. The output of this network will be a loss function $L(x_{m,1}, \dots, x_{m,n_m})$. Figure 3.1 presents a visual display of a forward pass through this computational graph. First, let's consider the case where there is only a single path in the DAG connecting $x_{1,k}$ to $L(x_{m,1}, \dots, x_{m,n_m})$. We can reference the vectors along this path p as $x_{p_1}, x_{p_2}, \dots, x_{p_l}$. We can compute the gradient of $x_{1,k} = x_{p_1}$ with respect to the loss function L, $\frac{\partial L}{\partial x_{p_1}}$ by taking the matrix product across the Jacobian matrices for every function along

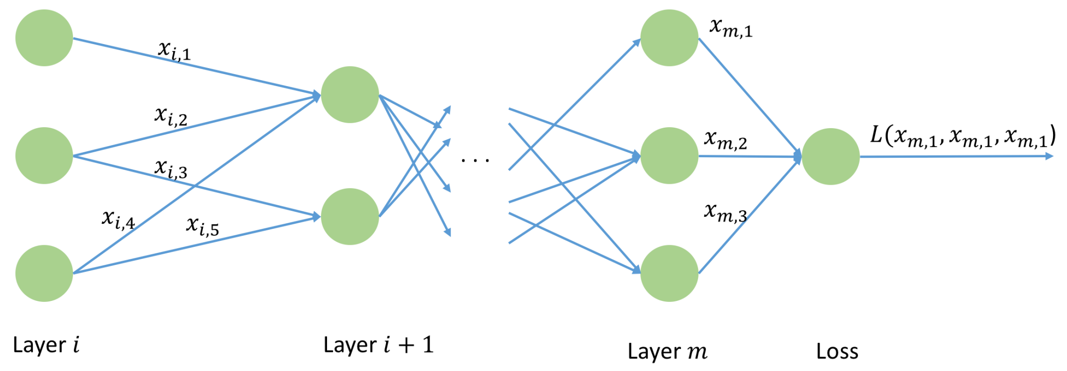


FIGURE 3.1: The computational graph representation of a forward pass through a typical neural network.

path p:

$$\frac{\partial L}{\partial x_{p_1}} = \frac{\partial L}{\partial x_{p_l}} \prod_{i=1}^{l-1} \frac{\partial x_{p_{i+1}}}{\partial x_{p_i}}$$

In the event that there are multiple paths, $p_1, \ldots, p_a$, between $x_{1,k}$ and $L(x_{m,1}, \ldots, x_{m,n_m})$, then the gradient of $x_{1,k}$ with respect to the loss function will be the sum of the matrix product of the Jacobians along each path:

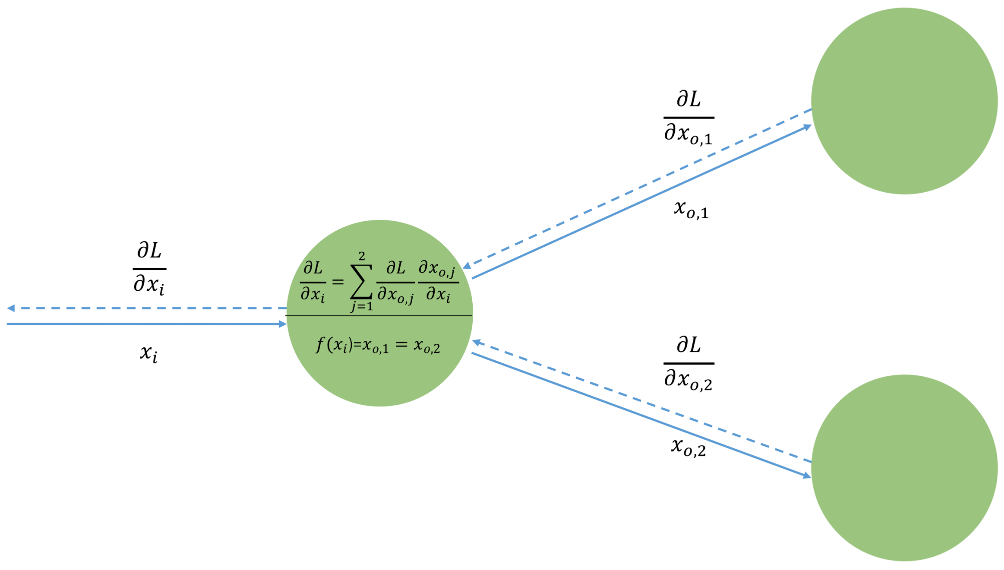
$$\frac{\partial L}{\partial x_{1,k}} = \sum_{j=1}^{a} \frac{\partial L}{\partial x_{p_{j,l_j}}} \prod_{i=1}^{l_j-1} \frac{\partial x_{p_{j,i+1}}}{\partial x_{p_{j,i}}}$$

This may at first seem like a large amount of computation for each backward pass, but many of the calculations required can be shared by computing gradients iteratively, layer by layer through the graph. By induction, as we reach each node of the DAG, we have already computed the gradient of the loss function with respect to each of the output vectors from this node, $x_{o,1}, \ldots, x_{o,b}$. Then, to compute the gradient of each input vector x_i to this node with respect to the loss function, we only need to sum the matrix products of the Jacobians $\frac{\partial x_{o,i}}{\partial x_i}$ with gradients $\frac{\partial L}{\partial x_{o,i}}$:

$$\frac{\partial L}{\partial x_i} = \sum_{j=1}^{b} \frac{\partial L}{\partial x_{o,j}} \frac{\partial x_{o,j}}{\partial x_i}.$$

Figure 3.2 visualizes a step in the recursive backpropogation algorithm described above.

The backpropogation algorithm described above allows us to efficiently compute



$\frac{\partial L}{\partial x_{o,1}}$
$x_{o,1}$
$\frac{\partial L}{\partial x_i}$
$\frac{\partial L}{\partial x_i} = \sum_{j=1}^{2} \frac{\partial L}{\partial x_{o,j}} \frac{\partial x_{o,j}}{\partial x_i}$
$f(x_i) = x_{o,1} = x_{o,2}$
x_i
$\frac{\partial L}{\partial x_{o,2}}$
$x_{o,2}$

FIGURE 3.2: The computational graph representation of one step of a backward pass through a typical neural network.

gradients of the parameters of a neural network model with respect to any differentiable scalar loss function of the outputs of the network. First-order optimization methods can be applied to these gradient values to set the parameters in the model so as to (locally) minimize the loss function.

In this chapter we will introduce GreedyNet, which is a method to embed the argmax function of certain optimization problems as a node in a neural network. We utilize GreedyNet to present SafetyNet, a neural network architecture which can be optimized with respect to a task-based loss function related to our omni-channel fulfillment problem. SafetyNet embeds a Threshold Policy as part of its internal state, allowing us to obtain high-quality Threshold Policies from an optimized instance of the model.

3.3 Related Work

Much of the mathematics underlying neural networks, including the backpropogation algorithm, has been known for decades, but in the past five years we have seen a resurgence of interest in these models as neural networks have been found to be highly effective in solving a wide range of machine learning tasks. Neural networks have been used to obtain state-of-the-art performance on many tasks including ones related to computer vision [40], natural language processing and translation [56], and game playing (Go, Atari) [51]. One driving force behind these recent advances has been the increasing availability of computing power for processing neural network operations. In particular, advances in Graphics Processing Units (GPUs) have made possible the fast matrix operations necessary to rapidly perform forward and backward passes through neural networks [13].

Of particular relevance to our research is previous work on argmin differentiation for various classes of optimization problems as a node in a neural network [26, 43]. A good example from this body of work is the recent development of OptNet [5], a neural network architecture that embeds the argmin function of a quadratic program inside a neural network. It is possible to implement our SafetyNet models using an OptNet layer. However, this adds unnecessary complexity to our model, and we demonstrate that when we can use a more efficient network layer that leverages the specific structure of the embedded optimization problem. Sensitivity analysis techniques in Operations Research and Operations Management [8, 23] are also closely related and may be used to efficiently embed certain optimization problems inside neural networks.

To the best of our knowledge, GreedyNet is the first attempt to embed differentiable argmin or argmax functions in a neural network using greedy algorithms. However, GPUs have previously been used in fast implementations of greedy algorithms for other purposes such as compressed sensing [7]. This line of work provides implementations similar to what we use in GreedyNet's forward pass. Backward passes of gradient information, however, are not considered in this prior work on GPU implementations of greedy algorithms.

3.4 GreedyNet: Differentiable Greedy Optimization

GreedyNet is a neural network architecture that allows the argmin or argmax function for certain optimization problems to be embedded in neural network layers. In this section we present the forward and backward pass algorithms of Greedynet, we state the requirements for an optimization problem to be compatible with GreedyNet, and lastly we prove the correctness of the forward and backward pass operations when these required conditions are met. In Section 3.7 we evaluate the running time of GreedyNet by comparing performance with OptNet [5] on an optimization problem compatible with both architectures.

3.4.1 Forward and Backward Pass Algorithms

We present the forward and backward pass algorithms of GreedyNet under the assumption that the input parameters are initialized with some form of randomness and consequently the probability of a tie between elements on the Scores function or that the final value of Fraction is exactly equal to zero or one is infinitesimal. This is a standard practice in neural network architectures, with the Rectified Linear Unit (ReLU) as a popular example. The ReLU function $f(x) = \max(0, x)$ is a commonly used non-linear activation function in neural networks, even though it is non-differentiable at zero. Implementations of ReLU neural network layers will typically return either zero or one as its derivative in the unusual event that the network attempts to evaluate its derivative at zero [34]. We will follow this example and have our code return a derivative of zero at isolated non-differentiable points for our implementation of GreedyNet. In the following discussion we will assume that our parameters are such that the Scores function does not have ties and the value of Fraction is in between 0 and 1.

Forward Pass

The solution vector z is initialized as a vector of all zeros. z is transformed one element at a time into the optimal solution based on an updating score vector, Scores. At each iteration, Scores is computed based on the current solution, z , and the input values, θ (Scores $=$ GreedyScore(θ, z)). The index of the maximum element in Scores, i, is recorded, and z_i is reassigned to 1. After each iteration a stopping criteria is calculated, and once this criteria is met none of the elements of z currently still at value 0 will be altered. When the stopping criteria is met, this often signals that the current solution is in violation of some constraint, and so the most recently updated element in z may be reassigned to the maximum fractional value between 0 and 1 such that the resulting vector z is feasible. Pseudocode for this forward pass algorithm is shown in Figure 3.3.

Backward Pass

An important observation is that an infinitesimal adjustment of the input parameters will only shift the value of the fractional element in solution z. This motivates the backward pass algorithm. During the backward pass, GreedyNet stores the value in index AddIndex of $\frac{\partial L}{\partial z}$[AddIndex]. Then, this value is multiplied element-wise with $\frac{\partial \text{FixViolation}(z,\theta,\text{AddIndex})}{\partial \theta}$ to obtain $\frac{\partial L}{\partial \theta}$.

3.4.2 Requirements and Correctness of GreedyNet

The most fundamental requirement for an optimization problem to be compatible with GreedyNet is that it must be solvable by a greedy algorithm. More specifically, we require that the optimal solution, represented as vector z, can be computed according to the forward pass algorithm of GreedyNet described in Section 3.4.1 and in Figure 3.3.

FixViolation$(z, \theta, \text{AddIndex})$ is the function that determines the magnitude of the final fractional value that gets assigned after the stopping criteria is met during the forward pass. The role of function FixViolation$(z, \theta, \text{AddIndex})$ may also be seen in the pseudocode shown in Figure 3.3.This function FixViolation$(z, \theta, \text{AddIndex})$ must be differentiable with respect to θ. Lastly, we require that the GreedyScore(θ, z) function must be continuous with respect to θ.

```
function FORWARD($\theta$)
    $z \leftarrow$ ZEROS($n$)
    Options $\leftarrow$ ONESLIKE($z$)
    while STOPPINGCRITERIA($z, \theta$) = False do
        Scores $\leftarrow$ GREEDYSCORE($\theta, z$)
        AddIndex $\leftarrow$ INDEXMAX(Scores $\circ$ Options)
        $z$[AddIndex] $\leftarrow 1$
        Options[AddIndex] $\leftarrow 0$
    end while
    Fraction $\leftarrow$ FIXVIOLATION($z, \theta$, AddIndex)
    $z$[AddIndex] $\leftarrow$ Fraction
    self.SaveForBackward $\leftarrow$ ($z, \theta$, AddIndex)
    return $z$
end function

function BACKWARD(self, dL_dz)
    $z, \theta$, AddIndex $\leftarrow$ self.SaveForBackward
    dL_dFraction $\leftarrow$ dL_dz[AddIndex]
    dFraction_dTheta $\leftarrow \frac{\partial \text{FIXVIOLATION}(z,\theta,\text{AddIndex})}{\partial \theta}$
    dL_dTheta $\leftarrow$ dL_dFraction $\circ$ dFraction_dTheta
    return dL_dTheta
end function
```

FIGURE 3.3: GreedyNet Forward and Backward Pass Pseudocode

Proposition 5. *The forward pass algorithm returns the optimal solution to the target optimization problem of GreedyNet.*

Proposition 5 is true by definition, as this is a requirement for an optimization problem to be compatible with GreedyNet. This forward pass algorithm describes a fairly general framework for a greedy algorithm. z represents a (possibly fractional) set of elements that define a solution. At each iteration a score for each element not in the set is calculated, and the item with the highest score is added added to the set. Between iterations, the stopping criteria checks if the algorithm is finished or may continue to add all or part of another element to the solution set. Before the algorithm stops it checks to see if the last element added to the set needs to be shifted down to a fractional value so the overall solution is feasible.

Proposition 6. *The backward pass algorithm returns the correct gradient* $\frac{\partial L}{\partial \theta}$.

Proof. We observe that an infinitesimal adjustment of parameters θ will only alter the value of the fractional element of z. This is the element at index AddIndex, which we will re-label as index i for this proof. This observation is true because of the assumption that the scoring function, GreedyScore(θ, z), is continuous with respect to θ. Without a tie in maximum scores, an infinitesimal adjustment to θ would not change the index of the final element added to z, only the magnitude of its fractional value. Mathematically, we express this observation as $\frac{\partial z}{\partial \theta}[\text{AddIndex}, k] = \frac{\partial \text{FixViolation}(z,\theta,\text{AddIndex})}{\partial \theta}[k]$ and $\frac{\partial z}{\partial \theta}[j, k] = 0, \forall j \neq \text{AddIndex}, \forall k \in 1, \ldots, m$.

We can obtain a closed-form expression for the gradient of the loss function, L, with respect to θ using the chain rule of calculus:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial \theta}.$$

$\frac{\partial z}{\partial \theta}$ is an $n \times m$ Jacobian matrix, the matrix of partial derivatives of a vector-valued function, and by our above observation, it is entirely zero except for its i-th row. Consequently, when we perform the matrix multiplication of $\frac{\partial L}{\partial z} \cdot \frac{\partial z}{\partial \theta}$ we find that $\frac{\partial L}{\partial \theta}[k] = \frac{\partial L}{\partial z}[i] \times \frac{\partial z}{\partial \theta}[i,k]$, $\forall k \in 1,\ldots,m$. We observed above that $\frac{\partial z}{\partial \theta}[i,k] = \frac{\partial \text{FixViolation}(z,\theta,i)}{\partial \theta}[k]$, and so the chain rule expression for the true gradient is equivalent to what is computed during the backward pass of GreedyNet.

□

3.5 SafetyNet: A Neural Network Model for Joint Estimation and Optimization of the Knapsack Threshold Model

We return to the practical problem of optimizing inventory exposure in omni-channel retailing using threshold policies. To this end we present SafetyNet, an artificial neural network architecture we use to compute threshold policies for the omni-channel fulfillment problem studied in Chapter 2. SafetyNet differs from the methods presented in Chapter 2 in that it all estimation of the model parameters is performed simultaneously with threshold optimization in a single end-to-end model. One concrete advantage of this paradigm over separate estimation and optimization pipelines is that in the event that estimation errors lead to threshold policies that miss the cancel rate target, SafetyNet is able to adjust the coefficients of an embedded Knapsack Threshold Problem linear program in an adaptive manner. SafetyNet's utility when applied to problems in omni-channel retailing motivates the development of GreedyNet (Section 3.4), as GreedyNet can be used to greatly improve the speed and scalability of SafetyNet implementations. We describe an overview of three variants of the SafetyNet model, then we describe each layer of the network in detail, followed by the algorithms we use to train SafetyNet.

3.5.1 Architecture Overview

There are two source nodes, Input Data and Generative Parameters, which begin any forward pass through the network. The input data is the batch of orders data, each order containing information on price, inventory level at time of order, item category, cancel status (whether the order was successfully filled or cancelled), and the threshold policy in place at the time of collection. The generative parameters can be interpreted as the current state of the network's internal parametric model

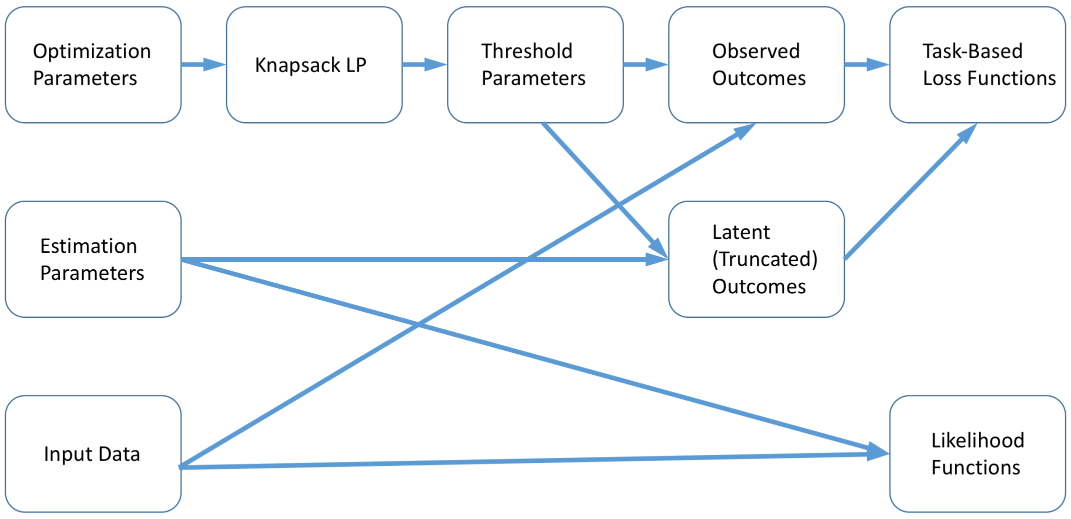


FIGURE 3.4: The architecture of SafetyNet, featuring a Knapsack LP layer that uses a differentiable linear program approximation and a Latent Outcomes layer that uses input data and generative parameters to correct data truncation.

of the un-truncated stream of orders data. They includes cancel logistic function parameters $\beta_{0,i}$, and $\beta_{1,i}$, inventory level distribution parameters λ_i and k_i, and demand distribution parameters d_i for each item category i. Typically, average prices p_i for each item category are fixed values known to the retailer, but it is also very easy to store these values as adjustable parameters. We store two copies of each of these parameters, one copy (optimization parameters) is used to formulate the linear program in the Knapsack LP layer, and the other (estimation parameters) is used to estimate the impact of data truncation on threshold policies produced by SafetyNet. The cancel rate target C is also a model parameter, though only one copy of this parameter is needed, for use in the Knapsack LP layer. The Knapsack LP layer contains a differentiable encoding of the linear programming relaxation of (2.1) which takes elements from Generative Parameters as inputs and outputs a threshold policy to the subsequent layers. Observed outcomes computes the revenue and cancels of the threshold policy from the Knapsack LP layer as applied to the batch of Input Data. Latent Outcomes computes an estimate of additional revenue and cancel outcomes related to orders that are filtered out of the Input Data due to truncation. Task-based Loss Functions return truncation-adjusted revenues and cancels, as well as log likelihoods of the Input Data with respect to the estimation version of the Generative Parameters.

3.5.2 SafetyNet Models

We focus on three versions of the SafetyNet model: Full SafetyNet, No-Optimization SafetyNet, and MLE SafetyNet.

Full SafetyNet

The Full SafetyNet model is the version of SafetyNet where backward passes through the network, in particular the Knapsack LP layer, are used to directly update stored elements in Generative Parameters. These updates result in the optimization parameters taking on values that may diverge from the estimation parameters. The elements of the right-hand side of the cancel constraint in LP (2.1) also update during backward passes through the network, allowing Full SafetyNet to find threshold policies that closely track the retailer's cancel rate requirements. The estimation parameters converge to truncation-adjusted maximum likelihood estimates while the optimization parameters adjust themselves to produce threshold policies that optimize the network's task-based loss functions. We discuss two versions of Full SafetyNet, where the Knapsack LP layer is implemented using either the GreedyNet architecture or the OptNet architecture [5]. In Section 3.7 we compare the running time performance of these two optimization layers. For the purpose of understanding the SafetyNet model the GreedyNet and OptNet implementations are largely interchangeable as they perform the same high-level operations during the forward and backward passes.

No-Optimization SafetyNet

This model removes the Knapsack LP layer and represents the threshold policy as a discrete distribution that are just additional parameters of the neural network model. During backward passes through the network, gradients of the task-based loss functions are propagated through the network so it can iteratively update the stored (probabilistic) threshold policy in an improving direction.

MLE SafetyNet

The MLE SafetyNet model sets the optimization parameters equal to the estimation parameters and uses backward passes through the network only to update the estimation parameters within the Generative Parameters with respect to the likelihood of the Input Data. The task-based outputs of the network such as revenue and cancel rates are not used to train MLE SafetyNet, and so the Generative Parameters simply reflect truncation-adjusted maximum likelihood estimates. This model is conceptually similar to our Expectation-Maximization model introduced in Section 2.6.2, though MLE SafetyNet is able to account for multiple threshold levels used during data collection. Our EM method assumes a single set of thresholds to collect its training data, which can lead to inaccurate estimates if this assumption is violated.

3.5.3 Generative Parameters

The Generative Parameters layer assumes a parametric model of the data where orders are samples drawn from discrete probability distributions over the item categories. The inventory level of each order is drawn from a Weibull distribution independently. The order status, specifically whether an order is cancelled is characterized by a logistic function over inventory counts conditioned on the item's category.

As mentioned earlier, two sets of parameters related to this model of the data are stored. One set is used to estimate the effect of data truncation and the other for the optimization of threshold policies. $d_E = [d_{1,E}, \ldots, d_{I,E}]$ and $d_O = [d_{1,O}, \ldots, d_{I,O}]$ represent that probability distributions of demands over item category. $(\lambda_{i,E}, k_{i,E})$ and $(\lambda_{i,O}, k_{i,O})$ define Weibull distributions of inventory positions

$$g_{i,E}(x) = e^{-(\frac{x}{\lambda_{i,E}})^{k_{i,E}}} - e^{-(\frac{x-1}{\lambda_{i,E}})^{k_{i,E}}}$$

and

$$g_{i,O}(x) = e^{-(\frac{x}{\lambda_{i,O}})^{k_{i,O}}} - e^{-(\frac{x-1}{\lambda_{i,O}})^{k_{i,O}}}$$

$\forall i \in I$. Similarly, $(\beta_{i,0,E}, \beta_{i,1,E})$ and $(\beta_{i,0,O}, \beta_{i,1,O})$ define logistic cancel functions

$$f_{i,E}(x) = 1 - \frac{1}{1 + e^{\beta_{i,0,E} + x\beta_{i,1,E}}}$$

and

$$f_{i,O}(x) = 1 - \frac{1}{1 + e^{\beta_{i,0,O} + x\beta_{i,1,O}}}$$

$\forall i \in I$. Each item category i has a price parameter p_i, and the cancel rate limit in the Knapsack LP is the parameter b_c. In No-Optimization SafetyNet, threshold policies are parameterized as a probability distribution where $t_{i,j}$ is the stored probability of setting the threshold for category i to inventory value j. The solution to the Knapsack LP layer in Full SafetyNet has the same representation as the threshold parameters of No-Optimization SafetyNet, but by Theorem 5 we are guaranteed that the solution can be made nearly-integral. The SafetyNet models are able to accommodate fully fractional threshold policies, but this is less desirable than integral or nearly-integral solutions from the perspective of a retailer who would want to implement these policies in production.

3.5.4 Knapsack LP Layer

The Knapsack LP layer embeds the linear programming relaxation of the Knapsack Threshold Problem integer program (Equations 2.1) as a neural network layer within SafetyNet. This embedding can be done with either a GreedyNet or an OptNet [5] layer. We show in Section 3.7 that GreedyNet is faster and scales to larger instances (more knapsack categories and threshold options) than can be solved by OptNet. Both GreedyNet and OptNet layers enable SafetyNet to solve this specific Knapsack Threshold Problem linear program during a forward pass of the network and allows gradients to be backpropogated through the linear program in this layer of SafetyNet.

We conclude in Section 3.7 that GreedyNet achieves better performance than OptNet within the context of implementing SafetyNet. However, some of our empirical results were obtained using SafetyNet models powered by OptNet so it is important to include its related implementation details along with these details for GreedyNet.

GreedyNet

GreedyNet was introduced in a more general setting in Section 3.4, so it is necessary to explain in detail how it can be applied to as the Knapsack LP layer of SafetyNet. The inputs to the forward pass, θ, will be marginal revenue and marginal cancel matrices which are computed directly from the stored SafetyNet model parameters. Element (i, j) of marginal revenue matrix M_r indicates the marginal effect on revenue from increasing the threshold on items in category i from $j-1$ to j. Similarly, element (i, j) of M_c, the marginal cancel matrix, indicates the marginal effect on cancelations from increasing this same threshold. These $n \times m$ matrices can be reshaped into vectors of length nm. We will refer to these vectors as v_r and v_c. The output vector z, which represents the threshold values, will also have nm elements.

The greedy scoring function GreedyScore(θ, z) used to compute scoring vector Scores is the vector of element-wise quotients of v_r and v_c. In other words, Scores$[i]$ = GreedyScore$(v_r, v_c) = \frac{v_r[i]}{v_c[i]}, \forall i \in 1, \ldots, nm$. We define our StoppingCriteria(z, θ) function as follows:

$$\text{StoppingCriteria}(z, v_c) = \left\{ \begin{array}{ll} 1 & \text{if } \sum_{i=1}^{nm} z[i]v_c[i] > C \\ 0, & \text{else} \end{array} \right\}.$$

The FixViolation$(z, \theta, \text{AddIndex})$ function computes the fractional value the last element added to the solution will get set to:

$$\text{FixViolation}(z, v_c, C, \text{AddIndex}) = 1 - \frac{\sum_{i=1}^{nm} z[i]v_c[i] - C}{v_c[\text{AddIndex}]}.$$

Theorem 8. *The forward pass of GreedyNet, as specified in this section, solves the linear programming relaxation of the Knapsack Threshold Problem.*

Proof. The solution vector z contains a representation of a threshold policy, but this may not be immediately clear. First, we will want to re-shape z into an $n \times m$ matrix we will call T_z. Element (i, j) of T_z indicates the probability an order for category i at inventory level j is accepted. For a policy defined by T_z to be a threshold policy, it must be the case that $T_z[i, j_1] \leq T_z[i, j_2]$ $\forall i \in [n]$, $\forall j_1, j_2 \in [m]$, $j_1 < j_2$. Fortunately, this will always be the case for solution matrices T_z. This is a consequence of the parameterization of the coefficients in the Knapsack Threshold Problem LP. The quotient of marginal revenue and marginal cancelations rate for category i at inventory level j is $\frac{(1-f_{i,O}(j))g_{i,O}(j)d_{i,O}p_i}{f_{i,O}g_{i,O}(j)d_{i,O}} = \frac{(1-f_{i,O}(j))p_i}{f_{i,O}(j)}$. Cancel rate function $f_{i,O}(j)$ is necessarily decreasing with respect to j and so the greedy scoring function is necessarily increasing within each category with respect to j. Therefore, T_z will always obey the required condition and represents a threshold policy.

We conclude the proof by arguing the threshold policy represented by T_z is an optimal fractional threshold policy. For this, we will use an argument commonly used to prove that a similar greedy algorithm is optimal for the linear programming relaxation of the classical Knapsack problem. Suppose there is another feasible solution that obtains more revenue than the policy represented by T_z. Let this solution be represented by T'. Then there must be i_1, i_2, j_1, j_2 such that $T'[i_1, j_1] > T_z[i_1, j_1]$ and $T'[i_2, j_2] < T_z[i_2, j_2]$. It is also necessarily true that $\frac{M_r[i_1,j_1]}{M_c[i_1,j_1]} \leq \frac{M_r[i_2,j_2]}{M_c[i_2,j_2]}$, otherwise $T_z[i_1, j_1] = 1$. This is a contradiction, proving the optimality of the threshold policy obtained by the forward pass of GreedyNet. □

The remaining requirements for using GreedyNet as a layer of SafetyNet are that FixViolation$(z, v_c, C, \text{AddIndex})$ is differentiable and GreedyScore$(v_r, v_c,)$ is continuous. Earlier in this same section we have provided closed-form expressions for these functions that clearly demonstrate these conditions are met.

We have also performed extensive empirical checks on the accuracy of both the forward and backward passes. We compare the threshold policies obtained from forward passes of GreedyNet with solutions to the original linear programming formulation and find that the mean value of the maximum difference of elements in solution vector z is less than $\frac{8}{10^7}$ across $10,000$ randomly generated instances. We also compare relative error the gradient for these same instances by a numerical gradient check and get a mean relative error less than $\frac{2}{10^6}$, confirming the accuracy of the gradients computed during backward passes of GreedyNet.

OptNet

OptNet is a neural network module implemented in PyTorch that embeds differentiable quadratic (and linear) programs into artificial neural networks, allowing gradients to backpropagate through the "arg max" function of the embedded optimization problem. This allows us to compute gradients of the underlying optimization parameters with respect to the task-based outcomes. We embed the same linear program (2.1) described in Section 2.4.1, with coefficients determined by differentiable functions of θ. In this LP $c_{i,j} = \sum_{x=j}^{U}(f_{i,O}(x)g_{i,O}(x)d_{i,O})$ and $r_{i,j} = p_i \sum_{x=j}^{U}((1 - f_{i,O}(x))g_{i,O}(x)d_{i,O})$. The output of this layer is a matrix that encodes a threshold policy. By Theorem 5, the solution to this LP will be almost entirely integral, so most rows of this thresholds matrix provide a one-hot encoding of a threshold policy. In the event that one category has fractional variables, SafetyNet will interpret this as a probabilistic policy. The retailer can easily turn this fractional solution into an integer threshold either by taking an average or using a heuristic.

3.5.5 Observed and Latent Outcomes

The threshold policy is applied to the batch of input data to determine revenue and cancel outcomes. These values are computed in the Observed Outcomes layer. SafetyNet computes the expectation of the distribution of orders that was truncated when the batch of input data was collected, conditional on the thresholds used during data collection, by performing operations that combine the input data with generative parameters. The Latent Outcomes layer computes these conditional distributions and applies the thresholds obtained from the Knapsack LP layer to this estimation of truncated demand to obtain associated revenue and cancel outcomes. Lastly the outputs of the Observed Outcomes layer and Latent Outcomes layer are aggregated into several loss functions corresponding to the objective function of the stochastic program we are attempting to solve and the violation of each of the constraints that must be satisfied in expectation.

The Latent Outcomes layer contains a series of calculations over the input data and stored parameters to obtain estimates about data that was truncated as the input data was collected. For each order o in the input data, we compute the estimated acceptance probability across all incoming orders of this category under the thresholds in place at the time of collection. This acceptance probability for category i orders collected under thresholds t (where t is the set of thresholds specific to this order o and $t_{i,j}$ is the probability the threshold for category i is inventory level j) is $\alpha(i,t) = \sum_{y=0}^{U} g_{i,E}(y) \sum_{x=0}^{y} t_{i,x}$. The expected quantity of orders truncated corresponding to order o is $m(i,t) = \frac{1}{\alpha(i,t)} - 1$. We then distribute these $m(i,t)$ estimated truncated orders across all inventory levels according to the estimated distribution of inventory levels. $m(i,t,j) = m(i,t)g_{i,E}(j)$ defines this distribution, where $m(i,t,j)$ is the expected quantity of truncated orders of category i at inventory level j corresponding to each order in the batch of category i received under threshold value t. The Latent Outcomes layer aggregates these values over all orders in the input data to obtain overall estimates of truncated orders for each category and inventory level that correspond to the input data. The thresholds received from the Knapsack LP layer are applied to these estimated truncated orders to determine estimates of revenues and cancels that will occur if these thresholds are applied to incoming orders.

3.5.6 Training SafetyNet

Optimizing the parameters in a large statistical model like SafetyNet is challenging, but we have identified methods tailored to the specific challenges of training SafetyNet that yield highly-effective threshold policies. The parameters in SafetyNet have an interpretation as a constructive model for the retailer's orders data, and we have found that maximum likelihood estimates serve as good initial values. In our experiments we use the benchmark set of thresholds (all threshold set to a single value, for example) to collect the first set of data and initialize all model parameters using this data.

We can use SafetyNet to find maximum likelihood estimates that account for data truncation for many of the model's parameters. Forward passes through SafetyNet return the likelihood of attributes of the input data, such as the counts of categories, inventory levels and cancel outcomes found in the inventory data. Backward passes starting from these likelihood-based outcomes allow us to find truncation-adjusted estimates of d_E, $\lambda_{i,E}$, $k_{i,E}$, $\beta_{i,0,E}$ and $\beta_{i,1,E}$, $\forall i \in I$.

During the first day of training we cut the Knapsack LP layer out of SafetyNet and run forward passes on batches of that day's order data starting from the input data and threshold variables taken from the benchmark policy. We use the backward passes through this network to obtain gradient estimates on the estimation parameters and update them by stochastic gradient descent. Before continuing to the second period of order collection we set the optimization parameters equal to their corresponding estimation parameters. Lastly, the cancel rate parameter C that appears on the right-hand side of the Knapsack LP are initialized to the observed cancel rate in the first day of data collected under the benchmark thresholds.

At all other times the thresholds used to control the flow of orders are those produced by SafetyNet. In Full SafetyNet and MLE SafetyNet, all forward passes begin with the optimization parameters and flow through the Knapsack LP layer to determine the threshold values. The difference between these models is that the optimization parameters get reset to the values of the corresponding estimation parameters after each iteration and C is not allowed to vary in MLE SafetyNet. No-Optimization SafetyNet removes the Knapsack LP layer and always operates with forward passes originating from the threshold distribution. Backward passes through No-Optimization SafetyNet update the threshold parameters directly.

On every forward pass, several outcomes are computed as output of the network: revenue, violation of cancel rate target, and likelihood measures of the data based on the estimation parameters. In all versions of SafetyNet the estimation parameters get updated by backward passes from the likelihood outcomes. Full SafetyNet and No-Optimization SafetyNet first checks if the violation of the cancel target is positive. If these targets are in violation the next update of the optimization parameters will use gradients obtained from backward passes originating from the relevant violation outcome. Otherwise, if all constraint violations are negative the network uses a backward pass from the revenue outcome to update the optimization parameters.

The Full SafetyNet model poses some unique challenges as backward passes backpropogate gradients through the arg max function of a linear program. This class of functions can be more sensitive to small changes than the non-linear activation functions typically seen in neural networks and we found that successfully optimizing the parameters of Full SafetyNet required using smaller learning rates for our parameter updates than were required for versions of the model without backward gradient passes through this arg max function. We that adaptive optimization algorithms such as Adam did not perform well on Full SafetyNet. These adaptive optimizers would either settle on solutions of very poor quality or would fluctuate between wildly different solutions every update. We suspect this is because SafetyNet produces multiple loss function outputs on each forward pass, and the values of these loss functions determine the origin of the backward pass used to produce the next gradient values. A consequence of this procedure is that loss function values used to initialize backpropagation can vary greatly between iterations, and this leads to the poor performance of the adaptive optimization algorithms we tested. Ultimately, we found that stochastic gradient descent with a small fixed step size allowed us to find effective parameter values for all versions of SafetyNet. Additionally, we found that for Full SafetyNet updating optimization parameters for a single category of each items performed better than updating all parameters on every iteration.

3.6 Empirical Results

3.6.1 Simulation Experiments

Methods

We use simulated data to evaluate how effective our models are at optimizing revenue while maintaining a pre-specified cancel rate in the presence of truncated data. We compare the performance of our six methods – Onera-SAA, Onera-MLE, Onera-EM, Onera-SN-Full, Onera-SN-NoOpt, and Onera-SN-MLE – for estimating the coefficients of this model to each other and to Retail-1-threshold policies which accept or reject orders for all items based on a single unified threshold. These Onera policies are implementations of each of the methods described in Sections 2.6 and 3.5: Sample Average Approximation, Maximum Likelihood Estimation, Expectation-Maximization, Full SafetyNet, No-Optimization SafetyNet, and MLE SafetyNet, respectively.

The distributions and parameters used in this simulation were selected to reflect the real data observed from a high-end multi-billion dollar fashion retailer. In Section 2.6 we demonstrated that our modeling choices are appropriate for this data. Each simulation takes place over 7 time periods on a set of 50 clusters (categories) of items. Note that at larger cluster numbers, the improved performance of our knapsack models will in fact be significantly more pronounced but even as few as 50 clusters is enough to make a significant impact. Demand for each cluster is modeled as a Poisson process whose rate parameter is drawn from a Normal distribution with mean 10 and standard deviation 5. Each day of the simulation allows the Poisson processes generating demand to run for either 40 or 400 time units (depending on simulation condition). Inventory distributions for each category are modeled as Poisson distributions with rate parameters drawn from a Uniform distribution from 1 to 20. The price of items in each category is drawn from a Normal distribution with mean 800 and standard deviation 400. Each order is assigned a cancel probability based on a logistic function of the form $1 - \frac{1}{1+e^{a+b \cdot I}}$, where I is the item's stated inventory level and a and b are parameters of the item's category. a and b are drawn for each category from Normal distribution with mean $.5$ and $-.2$ and standard deviations $.1$ and either $.05$ or $.15$ (depending on simulation condition), respectively. The inventory distribution parameters, cancel parameters, demand rates, and prices are all truncated at 0 to prevent negative values.

We run simulations of 8 scenarios that vary across the following conditions: amount of data truncation, volume of demand, and variability of cancel rate. We test two parameter options for each of these dimensions of variation. We vary data truncation by setting our benchmark Retail-1-threshold policies to single thresholds of either 8 or 14, allowing approximately $\frac{1}{3}$ and $\frac{2}{3}$ of total demand to get truncated, respectively. Volume of demand is varied by allowing either 40 or 400 time units for the demand generating Poisson process during each period of the simulation. We control cancel variability by running simulations where the standard deviation of cancel parameter b is $.05$ as well as $.15$. We test the Onera and Retail-1-threshold policies on 10 runs of each of the 8 (2^3) combinations of these simulation variants. We wish to compare against corresponding Onera policies. We use the corresponding Retail-1-threshold policy as the starting threshold for the Onera policies and then let the Onera policies train on the data it collects and determine its own thresholds for the remaining days. At the end of the simulation, we compare the Onera policy to the Retail-1-threshold policy in terms of cancel rate and revenues.

Results

We can visualize the differences in performance of our methods most clearly through scatterplots of the cancel rate and revenues of our policies on individual simulation runs. Figure 3.5 shows that across all conditions Full SafetyNet and No-Optimization SafetyNet most closely track the target cancel rate and provide increases in revenue that are along the efficient frontier of outcomes produced by our set of policies. We observe that the condition where truncation is high, demand low, and cancel variation high is the most challenging for our policies as there is less data available to learn from, a large amount of incoming demand is truncated by the benchmark policy and there is large variation in cancel rates between items. In contrast, results for the simulation with low truncation, high demand, and low cancel variation show the policies generally differ less from the benchmark, but again we see that Full SafetyNet and No-Optimization SafetyNet closely track the target cancel rate and provide a substantial increase in revenue.

In Figure 3.6, we present cancel rates, and revenue as percentage increases compared to the corresponding Retail-1-threshold policy in the following tables. The first three columns describe the truncation (T), demand (D) and cancel variation (V) conditions of each set of simulations, and the remaining columns present the revenue (and cancel) increases for each policy. Recall that the objective of our policies is to maximize the lift in revenue without increasing cancel rates.

We find that Full SafetyNet method yields the best performance of all Onera methods. Full SafetyNet averages 32.6% more revenue than its corresponding Retail-1-threshold policy across all simulations while increasing cancels only 1.2% on average, closely tracking the cancel rate target set by this benchmark. In general, we observe that the SafetyNet methods are able to track the target cancel rate set by the Retail-1-threshold policy more closely than the Separate Estimation and Optimization (SEO) methods, and this difference is magnified in the high truncation conditions. This suggests that SafetyNet's ability to account for data truncation gives it an advantage over the SEO methods that do not account for data truncation or in the case of Onera-EM account for truncation under assumptions that may be violated.

3.6.2 Real Data Experiments

Methods

It is also important to evaluate the performance of our algorithms using real retail data provided by Onera. We test the SafetyNet models introduced in this chapter using the same simulation as in Chapter 2. A strong motivating factor in developing the Full SafetyNet model is that it can adjust its internal optimization model to produce threshold policies that track the desired cancel rate. Consequently, an important measure of the utility of the SafetyNet models is whether they can produce threshold policies that accurately track the desired cancel rate while increasing revenues over the simple benchmark policy.

We compare the performance of the three SafetyNet models – Onera-SN-Full, Onera-SN-NoOpt, and Onera-SN-MLE – to each other, to a benchmark Retail-1-threshold policy, and to the results presented in Chapter 2 of the SEO models – Onera-SAA, Onera-MLE, and Onera-EM – in this simulation.

We will review the most important aspects of this experimental design. A complete description and discussion of the methodology used in this experiment can be found in Section 2.8.1. We used the 3 months of SFS orders from a multi-billion dollar annual online revenue fashion retailer to demonstrate our results. We evaluate our methods one week at a time, allowing our models to train on all data prior to the evaluation week, excluding the orders that are rejected by our artificial threshold

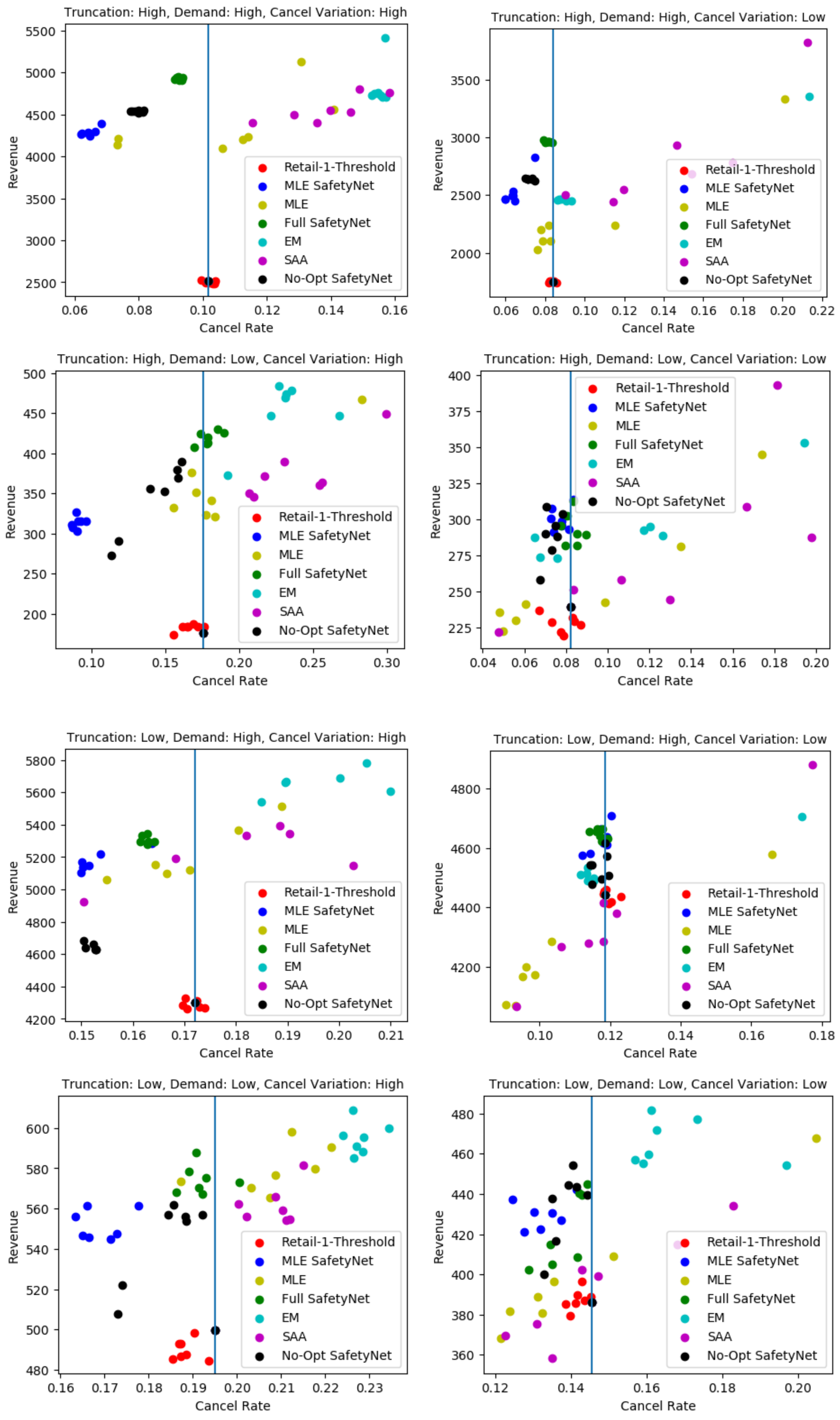


FIGURE 3.5: Performance comparison for individual simulations across conditions

Revenue (Cancel) Increase wrt Retailer Threshold					
T	D	V	Onera-SN-Full	Onera-SN-NoOpt	Onera-SN-MLE
High	High	High	91% (3%)	55% (-25%)	57% (-39%)
High	High	Low	25% (-1%)	18% (-5%)	21% (-7%)
High	Low	High	87% (5%)	65% (-9%)	52% (-41%)
High	Low	Low	7% (6%)	12% (0%)	12% (-3%)
Low	High	High	23% (-6%)	8% (-9%)	14% (-16%)
Low	High	Low	4% (-2%)	3% (-2%)	2% (-6%)
Low	Low	High	19% (3%)	13% (-5%)	14% (-14%)
Low	Low	Low	4% (2%)	4% (0%)	3% (-4%)

Revenue (Cancel) Increase wrt Retailer Threshold					
T	D	V	Onera-SAA	Onera-MLE	Onera-EM
High	High	High	76% (25%)	73% (9%)	90% (28%)
High	High	Low	23% (89%)	5% (38%)	27% (68%)
High	Low	High	77% (37%)	125% (4%)	91% (39%)
High	Low	Low	11% (125%)	0% (48%)	40% (117%)
Low	High	High	21% (1%)	20% (2%)	27% (5%)
Low	High	Low	2% (16%)	-4% (-8%)	6% (8%)
Low	Low	High	17% (9%)	17% (1%)	27% (20%)
Low	Low	Low	0% (11%)	-2% (-5%)	10% (16%)

FIGURE 3.6: Average outcomes by simulation condition

Simulation Results			
Policy	Cancels	Revenue	Increase
Retail-1-Threshold	6.19%	54.24%	N/A
Onera-SN-Full	5.27%	58.08%	7.07%
Onera-SN-NoOpt	5.53%	61.13%	12.69%
Onera-SN-MLE	4.83%	47.39%	-12.64%
Onera-SAA	7.09%	71.42%	31.65%
Onera-MLE	7.25%	75.75%	39.64%
Onera-EM	6.67%	72.29%	33.26%
Onera-Proxy	8.08%	78.52%	44.74%

FIGURE 3.7: Simulation results for JEO and SEO methods on real data

which is set to 8. We compare our Onera policies to a benchmark policy, Retail-1-Threshold, which sets a constant threshold of 10 to all orders. The target cancel rate for our models was 5.2%, which is somewhat lower than the cancel rate realized by this same policy across the weeks evaluated in the simulation, 6.2%. The target cancel rate is the realized cancel rate of the benchmark policy prior to the dates evaluated in this experiment. This discrepancy suggests that the real-world circumstances driving the data generation process may be changing over time.

Results

Figure 3.7 shows, for all policies tested, the mean cancel rates, mean revenues (as a percent of the maximum possible revenue), and percentage increase in revenue over the benchmark Retail-1-Threshold policy:

We find that the Full SafetyNet and No-Optimization SafetyNet models strictly dominates the benchmark policy in this experiment. It is also encouraging to see that the cancel rate realized by Full SafetyNet is very close to the target cancel rate (5.2%) and falls in between the target cancel rate and the realized cancel rate of the benchmark policy during the evaluation period. This result indicates that Full SafetyNet may be more robust to underlying changes in order patterns than the benchmark policy.

3.6.3 Discussion

The combined results from both sets of experiments, using simulated and real retail data, show that across all policies tested Full SafetyNet is able to closely track its target cancel rate while also providing a substantial increase in revenues over the benchmark policy. Revenues increased relative to the benchmark revenue from 4% to 91% in simulated data experiments and 8% in real data experiments using Full SafetyNet policies.

The difference between Full SafetyNet (Onera-SN-Full) and the SEO EM method (Onera-EM), the most similar of the SEO methods, is quite dramatic when we consider the cancel rates realized by these policies. In the simulated data experiments, Onera-EM exceeds the target cancel rate by at least 5% and by as much as 117% across the simulation conditions tested. In constrast, the maximum increase in cancel rate across all simulation conditions for Onera-SN-Full is 6%, and five of the other conditions tested had the realized cancel rate differ from the target rate by at most 3%. To be clear, these are discussions of percentage changes relative to the benchmark, which is why a 117% increase in cancel rate is possible.

A major difference between the Full SafetyNet and SEO EM models is that Full SafetyNet is able to adjust the embedded optimization problem based on the data observed. One specific adjustment Full SafetyNet is capable of making is decreasing the right-hand side value of its Knapsack LP in the event that too many orders are cancelling. It's important to know if this feature is the primary driver of the empirical success of Full SafetyNet or if the parametric model that formulates this LP is valuable. Note that the same parametric model is used for both Full SafetyNet, Onera-EM, and Onera-MLE. It is the process for estimating the underlying parameters that varies between these models. No-Optimization SafetyNet shares the adaptivity of Full SafetyNet but does not incorporate this parametric model. The improved overall performance of Full SafetyNet over No-Optimization SafetyNet supports the idea that using parametric models to construct the threshold-setting optimization problem is a valuable component of the Full SafetyNet model.

3.7 GreedyNet Performance Evaluation

In Section 3.5.4 we describe how both GreedyNet and OptNet architectures can be used to accurately implement the forward and backward passes required of the Knapsack LP layer in Full SafetyNet. Now we will compare the performance of these two architectures for performing the forward pass operation of solving the linear programming relaxation of the Knapsack Threshold Problem followed by a backward pass operation. We tested both architectures on randomly generated instances of this linear program at each of a variety of instance sizes. We scale the number of knapsack categories and the number of threshold options simultaneously, so an instance with size parameter k has both k knapsack categories and k threshold options. 100 randomly generated instances are evaluated for each of size parameters 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100. A comparison of the running times of GreedyNet and OptNet across these simulated conditions if plotted in Figure 3.8

We observe a dramatic difference between these architectures, with GreedyNet achieving much faster solve times. This is not surprising as OptNet is able to solve a wider range of optimization problems, but these results show that there is great benefit in SafetyNet from using GreedyNet layers. At the largest instance size tested (100 categories and 100 threshold options), GreedyNet took .82 seconds per iteration

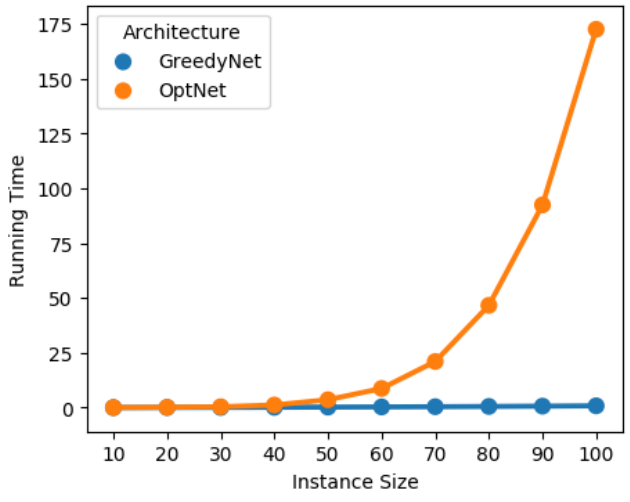


FIGURE 3.8: Running time comparison (in seconds) of GreedyNet and SafetyNet across varied instance sizes

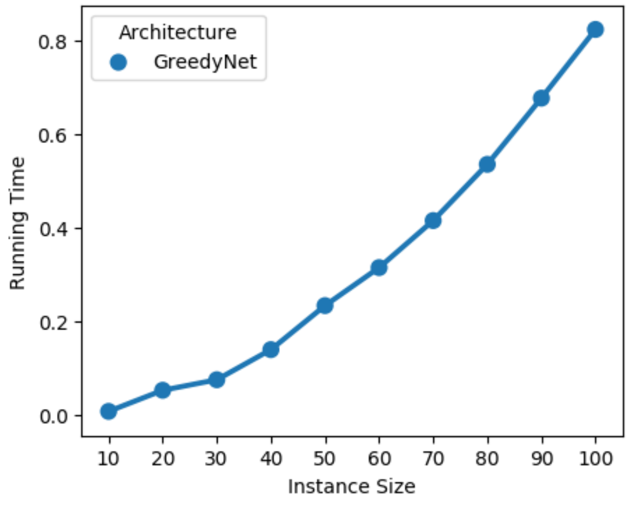


FIGURE 3.9: Running time (in seconds) of GreedyNet across varied instance sizes

on average, compared to an average running time of 172.46 seconds with OptNet. Even at the smallest instance size of 10 categories and threshold options, GreedyNet took an average of .008 seconds compared to an average of .019 seconds per iteration using OptNet. Figure 3.9 plots the running times of GreedyNet only to show that GreedyNet does slow down as the instance size increases, just at a different scale than we observe with OptNet.

3.8 Conclusion

In this chapter we introduce SafetyNet models to provide greater sophistication and more flexible solutions to the problem of efficiently limiting cancellations in ship-from-store omni-channel retail programs. We demonstrate on both real and simulated data that these methods are able to produce policies that dominate our benchmark policies and more closely track pre-specified cancel rate targets than the separate estimation and optimization methods introduced in Chapter 2.

We develop GreedyNet during this process as a tool to efficiently implement our SafetyNet models in a scalable way that would not be possible with pre-existing methods. For the Knapsack Threshold Problem we observe that GreedyNet provides a dramatic improvement over OptNet in running time and scalability. An open avenue of research is to find other applications for GreedyNet or similar neural network architectures that internally solve combinatorial problems. Some applications of GreedyNet could use a greedy algorithm as a heuristic for a more complex decision problem, allowing for complex planning to occur inside of a machine learning algorithm. Stochastic or online job scheduling may be an area where these techniques could be applied successfully.

Acknowledgments: Prof. R. Ravi and Dr. Srinath Sridhar are co-authors on this work. The real retail data analyzed in this chapter was graciously provided by Dr. Sridhar and his company Onera.

Chapter 4

Conclusion

Each chapter of this dissertation investigates different ways algorithms and mathematical models can be applied to alleviate some of the operational challenges faced by omni-channel retail businesses. In Chapter 1 I utilize linear programming sensitivity analysis to optimize fulfillment policies that fill as many orders as possible while accounting for location-specific shipping costs as well as probabilistic cancellation costs. Chapter 2 further studies the tradeoff between limiting cancelled orders and maximizing revenue in omni-channel ship-from-store program. I explore the effect of data truncation on this process and use methods from machine learning to overcome this challenge. Chapter 3 continues this line of work, introducing a joint estimation and optimization model that can adaptively track target cancel rates.

My goal in each of these chapters is to present results that are of interest to practitioners in the retail industry as well as researchers at universities. It is my hope that I have provided constructive insights into the conditions that are favorable for various fulfillment policies and the useful methods to account for data truncation in omni-channel fulfillment. It is also my intention for my technical contributions on differentiable optimization in neural networks to lead to advances in other application areas and to communicate meaningful connections from my models of omni-channel fulfillment to other classical models of Operations Research and Operations Management.

Bibliography

[1] Deepak Agarwal et al. "Personalized click shaping through lagrangian duality for online recommendation". In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2012, pp. 485–494.

[2] Alp Akcay, Bahar Biller, and Sridhar Tayur. "Improved inventory targets in the presence of limited historical demand data". In: *Manufacturing & Service Operations Management* 13.3 (2011), pp. 297–309.

[3] Seeking Alpha. *Lowe's Reports Fourth Quarter Sales And Earnings Results*. 2016. URL: https://seekingalpha.com/pr/16757192-lowes-reports-fourth-quarter-sales-earnings-results.

[4] Kareem Amin et al. "Budget optimization for sponsored search: Censored learning in MDPs". In: *arXiv preprint arXiv:1210.4847* (2012).

[5] Brandon Amos and J Zico Kolter. "OptNet: Differentiable Optimization as a Layer in Neural Networks". In: *arXiv preprint arXiv:1703.00443* (2017).

[6] Ravi Anupindi, Maqbool Dada, and Sachin Gupta. "Estimation of consumer demand with stock-out based substitution: An application to vending machine products". In: *Marketing Science* 17.4 (1998), pp. 406–423.

[7] Jeffrey D Blanchard and Jared Tanner. "GPU accelerated greedy algorithms for compressed sensing". In: *Mathematical Programming Computation* 5.3 (2013), pp. 267–304.

[8] J Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[9] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[10] Igor V Cadez et al. "Maximum likelihood estimation of mixture densities for binned and truncated multivariate data". In: *Machine Learning* 47.1 (2002), pp. 7–34.

[11] Kyle Cattani et al. "Boiling frogs: Pricing strategies for a manufacturer adding a direct channel that competes with the traditional channel". In: *Production and Operations Management* 15.1 (2006), p. 40.

[12] Li Chen and Adam J Mersereau. "Analytics for operational visibility in the retail store: The cases of censored demand and inventory record inaccuracy". In: *Retail Supply Chain Management*. Springer, 2015, pp. 79–112.

[13] Dan C Ciresan et al. “Flexible, high performance convolutional neural networks for image classification”. In:

[14] Kalyanmoy Deb, Karthik Sindhya, and Jussi Hakanen. “Multi-objective optimization”. In: *Decision Sciences: Theory and Practice*. CRC Press, 2016, pp. 145–184.

[15] Nicole DeHoratius and Ananth Raman. “Inventory Record Inaccuracy: An Emprical Analysis”. In: *Management Science* 54.4 (2008), pp. 627–641.

[16] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the royal statistical society. Series B (methodological)* (1977), pp. 1–38.

[17] Kris Johnson Ferreira, Bin Hong Alex Lee, and David Simchi-Levi. “Analytics for an online retailer: Demand forecasting and price optimization”. In: *Manufacturing & Service Operations Management* 18.1 (2015), pp. 69–88.

[18] Abercrombie Fitch. *Abercrombie Fitch's omnichannel sales grow 7.8 pct in Q1*. 2017. URL: https://www.digitalcommerce360.com/2017/05/25/abercrombie-fitchs-omnichannel-sales-grow-7-8-q1/.

[19] Jérémie Gallien et al. “Initial shipment decisions for new products at Zara”. In: *Operations Research* 63.2 (2015), pp. 269–286.

[20] Fei Gao and Xuanming Su. “Omnichannel retail operations with buy-online-and-pick-up-in-store”. In: *Management Science* (2016).

[21] Fei Gao and Xuanming Su. “Online and offline information for omnichannel retailing”. In: *Manufacturing & Service Operations Management* 19.1 (2016), pp. 84–98.

[22] Andres Garro. “New product demand forecasting and distribution optimization: a case study at Zara”. PhD thesis. Massachusetts Institute of Technology, 2011.

[23] Paul Glasserman and Sridhar Tayur. “Sensitivity analysis for base-stock levels in multiechelon production-inventory systems”. In: *Management Science* 41.2 (1995), pp. 263–281.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[25] Dick's Sporting Goods. *DICK'S Sporting Goods Reports Fourth Quarter and Full Year 2016 Results*. 2017. URL: http://www.prnewswire.com/news-releases/dicks-sporting-goods-reports-fourth-quarter-and-full-year-2016-results-300418947.html.

[26] Stephen Gould et al. “On differentiating parameterized argmin and argmax problems with application to bi-level optimization”. In: *arXiv preprint arXiv:1607.05447* (2016).

[27] Aravind Govindarajan, Amitabh Sinha, and Joline Uichanco. "Inventory Optimization for Fulfillment Integration in Omnichannel Retailing". In: (2017).

[28] Rupesh Gupta et al. "Email Volume Optimization at LinkedIn". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 97–106.

[29] Pavithra Harsha, Shivaram Subramanian, and Joline Uichanco. *Omni-channel revenue management through integrated pricing and fulfillment planning*. Tech. rep. Working Paper, Ross School of Business, University of Michigan, 2016.

[30] Yale T Herer and Ayelet Rashit. "Lateral stock transshipments in a two-location inventory system with fixed and joint replenishment costs". In: *Naval Research Logistics (NRL)* 46.5 (1999), pp. 525–547.

[31] Yale T Herer, Michal Tzur, and Enver Yücesan. "The multilocation transshipment problem". In: *IIE transactions* 38.3 (2006), pp. 185–200.

[32] Dorit S Hochbaum. "A nonlinear knapsack problem". In: *Operations Research Letters* 17.3 (1995), pp. 103–110.

[33] Shinji Ito and Ryohei Fujimaki. "Large-Scale Price Optimization via Network Flow". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3855–3863.

[34] Justin Johnson. *Simple examples to introduce PyTorch*. `https://github.com/jcjohnson/pytorch-examples`.

[35] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Berlin Heidelberg, 2013. ISBN: 9783540247777. URL: `https://books.google.com/books?id=wmL2BwAAQBAJ`.

[36] Sujin Kim, Raghu Pasupathy, and Shane G Henderson. "A guide to sample average approximation". In: *Handbook of Simulation Optimization*. Springer, 2015, pp. 207–243.

[37] Sujin Kim, Raghu Pasupathy, and Shane GForbes Magazine Henderson. "Information Acquisition and Exploitation in Multichannel Wireless NetworksOnline Sales To Boost Revenue For Urban Outfitters". In: (May 2017).

[38] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[39] Anton J Kleywegt, Alexander Shapiro, and Tito Homem-de Mello. "The sample average approximation method for stochastic discrete optimization". In: *SIAM Journal on Optimization* 12.2 (2002), pp. 479–502.

[40] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[41] Retsef Levi, Georgia Perakis, and Joline Uichanco. "The data-driven newsvendor problem: new bounds and insights". In: *Operations Research* 63.6 (2015), pp. 1294–1306.

[42] Retsef Levi, Robin O Roundy, and David B Shmoys. "Provably near-optimal sampling-based policies for stochastic inventory control models". In: *Mathematics of Operations Research* 32.4 (2007), pp. 821–839.

[43] Julien Mairal, Francis Bach, and Jean Ponce. "Task-driven dictionary learning". In: *IEEE transactions on pattern analysis and machine intelligence* 34.4 (2012), pp. 791–804.

[44] Adam J Mersereau. "Demand estimation from censored observations with inventory record inaccuracy". In: *Manufacturing & Service Operations Management* 17.3 (2015), pp. 335–349.

[45] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[46] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12.Oct (2011), pp. 2825–2830.

[47] Nicholas C Petruzzi and Maqbool Dada. "Pricing and the newsvendor problem: A review with extensions". In: *Operations research* 47.2 (1999), pp. 183–194.

[48] Evan L Porteus. "Stochastic inventory theory". In: *Handbooks in operations research and management science* 2 (1990), pp. 605–652.

[49] Forrester Report. *Why Every Online Retailer Should Ship-from-Store*. 2014.

[50] Total Retail. *Top 100 Omnichannel Retailers*. 2017. URL: http://www.mytotalretail.com/resource/top-100-omnichannel-retailers/file/.

[51] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7587 (2016), pp. 484–489.

[52] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *International conference on machine learning*. 2013, pp. 1139–1147.

[53] Tijmen Tieleman and Geoffrey Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude". In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.

[54] Aad W Van der Vaart. *Asymptotic statistics*. Vol. 3. Cambridge university press, 2000.

[55] J Wang and W Zhang. "Bid-aware gradient descent for unbiased learning with censored data in display advertising". In: *ACM Knowledge Discovery and Data Mining (KDD)*. Vol. 22. Association for Computing Machinery (ACM). 2016, pp. 665–674.

[56] Yonghui Wu et al. “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144* (2016).