Inbound Container Queuing Optimization Model For Distribution Centers

by

Russell G. Forthuber, PE

B.S. Marine Engineering Systems, United States Merchant Marine Academy, 2009

Submitted to the MIT Sloan School of Management and the Department of Mechanical Engineering in partial fulfillment of the requirements for the degrees of

Master of Business Administration
and
Master of Science in Mechanical Engineering

In conjunction with the Leaders for Global Operations Program at the
Massachusetts Institute of Technology
June 2017

Signature of Author _____ Signature redacted _____

MIT Sloan School of Management
Department of Mechanical Engineering
May 12, 2017

Certified by _____ Signature redacted _____

Dr. Stephen Graves,
Thesis Supervisor, Professor of Management Science
MIT Sloan School of Management

Certified by _____ Signature redacted _____

Dr. Stanley B. Gershwin,
Thesis Supervisor, Senior Research Scientist
Department of Mechanical Engineering

Accepted by _____ Signature redacted _____

Maura Herson,
Director of MBA Program, MIT Sloan School of Management

Accepted by _____ Signature redacted _____

Dr. Rohan Abeyaratne,
Chair of the Committee on Graduate Students
Department of Mechanical Engineering

This page intentionally left blank

Inbound Container Queuing Optimization Model For Distribution Centers

By

Russell G. Forthuber, PE

Submitted to the MIT Sloan School of Management and the Department of Mechanical Engineering on May 12, 2017 in partial Fulfillment of the Requirements for the Degrees of Master of Business Administration and Master of Science in Mechanical Engineering.

Abstract

Large, multi-national retailers have massive, worldwide supply chain networks which move product from a supplier to the end consumer. During the product's transit from a factory to a regional distribution center, customers may change or cancel their order, or the planned arrival date of the product at the distribution center may change. These products are packed in containers and arrive at the distribution center daily. Each day, humans may make decisions of which containers will be received at a distribution center and there are opportunity costs associated with selecting the wrong container to receive, namely, that the distribution center will become filled with product which is not immediately needed to meet outbound demand.

This thesis analyzes one method of receiving containers at a distribution center and the impacts it has on satisfying customers' orders. A model for a lean inventory management system and a container selection optimization model are described in it. Representative data is presented and the model is used to solve which containers should be received. Finally, the efficacy of the model and a comparison to a heuristic are discussed.

Dr. Stephen Graves
Professor of Management Science, MIT Sloan School of Management
Thesis Supervisor

Dr. Stanley Gershwin
Senior Research Scientist, MIT Department of Mechanical Engineering
Thesis Supervisor

Acknowledgments

I'd like to thank everyone at Nike for the opportunity to work on this project, the hours that were spent on answering questions were invaluable for this thesis and my professional development. Specifically, I'd like to thank Jason Trusley, Sandy Zuroff, Hugo Mora, Charlie Baker, Troy Sliter, Jessie Wakefield, Lee Beard, and Kristen Hanson for their support and time.

My advisors, Professors Stephen Graves and Stan Gershwin, were a great sounding board when I had doubts or questions with the direction of the project; thank you for your guidance.

A big thank you to my LGO classmates, LGO alum, and LGO staff for their insights, support, feedback, assistance, and fun times during the two years I've studied at MIT. This has been a transformational experience and I look forward to continuing to build bonds with you all in the future.

Finally, to my family and friends - thank you for helping me reach my goals and for your support when I come up short. The activities and experiences we've shared together through the past 30 years have turned me into who I am today. You've provided direction in life when I've needed it and equally as important accompanied me mountain biking, surfing, skiing, climbing, and whitewater kayaking- activities which have brought meaning to my life. While your contributions to this thesis and my degrees may be indirect, they are certainly present by keeping me motivated and driving forward.

Table of Contents

List of Figures

1 Introduction

The intent of this thesis is to present one form of an inbound container queuing optimization engine for application at a Distribution Center (DC); its logic is applicable to any DC or facility which exhibits similar structure to a DC.

The program's purpose is to take a larger set of inbound containers available for receipt at a DC and, given a receiving capacity constraint which is equal to or less than the entire set of containers, selects the best container subset to receive. It does this by taking outbound sales orders from the DC, inventory in the DC, and products within the container set at large into consideration, then an integer optimization computer program is run to compute the container subset to receive. In this thesis, the act of receiving a container is defined as the process of unloading all the product inside the container. The research for this thesis was conducted during the author's six month internship at Nike, Inc (Nike).

2 Overview of Nike

Nike was formed in 1964 on a handshake between co-founders Bill Bowerman and Phil Knight. The company was originally called Blue Ribbon Sports and was an importer of Japanese running shoes to the US market. As the company evolved, it has become the world's largest sportswear design, development, and manufacturing company.[1]

The Nike "Swoosh" logo is recognized across the world as a brand which serves athletes with premium quality product. Nike's revenue in 2015 was USD $30 billion with plans to grow to USD $50 billion by 2020. Its mission is to "bring inspiration and innovation to every athlete* in the world. *If you have a body, you are an athlete." The company is passionate about sport, environmental sustainability, and community impact. Subsidiaries of Nike include Hurley, Converse, Jordan, and Nike Golf.

Nike manufactures its product through contract manufacturers in over 600 factories spread through more than 40 countries. Product is then transported via vessel, air, truck, and/or rail through an intermodal transportation system to its DC's. These DC's serve regional retail customers (e.g. Dick's Sporting Goods or Footlocker) as well as its direct to consumer (DTC) market. Product flowing through this supply chain consists of build to order (also known as futures orders) as well as build to stock (also known as "Always Available" or simply AA) product.

[1] Phil Knight, "Shoe Dog"

3 Project Methodology

The format of this thesis will be in the same manner as the methodology taken during the project. The first part begins with background information pertaining to a typical large, multi-national retailer's supply chain, followed by a problem statement, a description of the current state, an understanding of the goals which the current state is attempting to accomplish (and therefore the basis of the solution), and finally a formulation of a proof of concept.

3.1 Investigation

The investigation into the background information and current state relied primarily on interviews, presentation decks, and analyzing data to generate a full understanding of the situation and all its nuances. As the author's understanding of the relationship between the problem and the current state became clearer, an iterative approach was used to develop a proof of concept solution to meet the needs of the company in a better way than the current state meets those needs.

An important aspect of this investigation is visiting the DC and understanding how front line workers perform their work at the DC. This is critical to rectify misunderstandings of how headquarters and DC prioritized containers. Headquarters and a DC may prioritize containers to be received during a day differently - headquarters takes a strategic approach to selecting containers for receipt, but the DC must take a tactical approach dependent on which containers are physically in the DC yard ready for receipt. Visiting the place where the work is actually done is important to create a comprehensive container queuing system which meets both the strategic and tactical needs of any large multi-national retailer.

3.2 Problem Formulation

After investigating the current method of selecting containers to receive each day, understanding the goals of the business with its receiving process, and the constraints a solution must take into account, formulating a mixed integer linear program was determined to be the best approach to solve the problem. The formulation of the problem may require several iterations as business needs become clear or are changed.

3.3 Proof of Concept and Testing

For this thesis, the lean inventory model was created using Alteryx, but the optimization engine proof of concept was coded in a program called Gurobi using Python and demonstrated for use to stakeholders. Performance modeling of the program against the current method of container selection was not done because of time constraints. However, the mixed integer optimization program is demonstrated with fake

data in Section 9 and a discussion of the results against a heuristic is included in Section 10.

4 Literature Review

The literature review in this thesis is important to help the reader understand the science behind why the model is formulated as it is. The information provided here is a brief overview of the theories used in the model formulation and if more information is needed, the references can be reviewed in further depth. By understanding the material in this chapter, the reader is in a position to fully understand both the gaps in the current container receiving methodology, and the logic of the solution model's formulation.

Several theses from MIT LGO graduates were reviewed before heading to the location and during the writing of this thesis. One particularly important thesis for the literature review was "Size Curve Optimization for Replenishment Products" by Andrew Gabris, LGO '16. In addition, other useful supply chain management and math sources were used in this literature review and the formulation of the solution.

4.1 Safety Stock

The purpose of safety stock is important to understand when developing a lean inventory model which has build to stock product in it. In a build to stock model, customers order product in a somewhat unpredictable, indeterministic manner. A producer either forecasts the amount of product to be consumed during a time period, or a producer orders what has been consumed over the past time period since the prior replenishment period plus lead time for product to travel through the supply chain. In either case, the producer does not want to experience a stockout due to the costs associated with lost sales.

A safety stock acts as a buffer against this stockout possibility and is calculated using a customer service level desired by the producer. The safety stock level is then added to the estimated inventory consumption for a consumption period so that the producer expects to end that period at the safety stock level. This buffer not only buffers against the unpredictability in consumer purchasing behavior, but also buffers against "supply-time variability and process lead-time variability".[2]

4.2 Linear and Integer Optimization Programs

In the context of this thesis, linear optimization programs and integer optimization programs are mathematical programs which are used to find solutions to complex analytical problems. These models may have hundreds to thousands of potential outcomes which the program solves using algorithms. Optimization programs can be used for

[2] Willems, "Inventory Optimization: Evolving from Fad to Necessity" 13

such things as a factory deciding how much product to source from various raw materials manufacturers, or scheduling of works for shifts.

The components of a linear optimization program are a *linear objective function*, *decision variables*, and *linear constraints.*[3] The objective function is the output of the program and the program's goal is to find a maximum or minimum value for this objective function. The decision variables are the levers which are adjusted in order to drive the objective function towards the maximum or minimum, while the constraints bound the decision variables to ensure the solution is limited to a feasible region where an optimal solution may exist. The constraints may be inequalities or equalities.

An example use case of a linear program could be the sourcing of raw materials for making final products at a factory. There may be many sources for the raw materials, each with different cost rates based on weight for their raw materials, varying quality of the raw materials, transportation costs to get to the factory, and maximum capacities which they will sell to the factory. The objective function of the factory may be minimize costs, the decision variables may be the mass of raw material bought from each source, and the constraints may be the average material quality must be of a certain purity, each sourcing location can supply only a certain number of tons, etc...

There are a number of algorithms a computer program can use to solve such a problem, but discussion of the types of algorithms and how they solve such a problem is outside the scope of this literature review. If a feasible solution exists and the program is solved, each decision variable will take the form of a continuous variable stating the exact mass of raw materials to be sourced from each source, and the objective function will show the total cost of this sourcing strategy. The fact the output is a continuous variable is an important characteristic of linear optimization programs - linear programs are easier to solve than integer programs.

An integer program[4] is formulated in the same way as a linear program except that it adds the additional constraint that at least one of the decision variables must be an integer. This is useful when looking for a discrete, rather than continuous, solution to a problem. Using binary values of 1 and 0 is an especially useful form of an integer program as it opens up more problems to be solved by optimization. Examples of integer programs using binary values as "yes/no" values are capacity planning - whether or not to build a factory in certain locations, or the decision to receive a container or not.

---

[3] Burke et al. *Search Methodologies* 69-70

[4] Bradley, Hax, Magnanti. *Applied Mathematical Programming.* 366-397

The challenge associated with integer programs is that it's more difficult to find an optimal solution than a linear program due to the fact there are discontinuities in the bounds of the constraints. A linear program's algorithm can make use of the fact that the constraints are linear and continuous, and that the optimal solution lives on the boundary of the feasible set. An integer program's algorithm requires individual calculations to be performed, which, for a binary problem with n decision variables, means there can be up to 2^n calculations which need to be performed. This can take a considerable amount of time.

4.3 TPS & Lean Production Management[5]

The Toyota Production System, or TPS, is synonymous with lean production. It focuses on limiting waste (or *muda*) in an organization and there are seven types of waste defined within TPS: overproduction, waiting, unnecessary transport, over-processing, excess inventory, rework, excessive motion.

The above forms of waste actually destroy value in a product because time and resources are spent on these when they aren't necessary, causing the cost of goods sold (COGS) to increase while the value proposition to the customer remains the same. From this mindset of eliminating waste comes the concept of a pull system to synchronize work.

A pull system uses customer demand as a signal to begin work. With a modest amount of inventory in processes, the customer creates a demand for product with the system responding by processing the work for that demand. It is the consumption of a specific set of parts downstream, or demand for those parts, as a trigger to release more units for work. Such a strategy prevents additional inventory from remaining in the process.

For a supply chain with short lead times, his type of supply chain strategy is useful for meeting the immediate demands of a customer and keeping inventory to a minimum. A shortcoming of this strategy is in a supply chain with long lead times because a customer's demand will not be met in a timely manner if the demand signal is needed to begin production.

A push supply chain pushes product through the supply chain to the customer. The quantity of product produced and is typically done using forecasts or prior ordering history with a customer. The push supply chain does not respond well to quick changes in an end customer's demand because of its nature. However, if there is sufficient supply of product in the system, the quantity of product typically meets a customer's demand.

---

[5] Terwiesch, *Matching Supply With Demand* 202-206

There are pros and cons of each system for a customer and company. One way to blend the pros and cons is to create a push-pull supply chain, where product is produced and shipped using a push strategy, with a node being used as a transition point to turn the push supply chain into a pull supply chain. The location of that node in this thesis is at the DC where containers are selected for receipt and inventory is held inside to meet customer's demand.

5 Large, Multi-National Retailer Supply Chains

Describing a large multi-national retailer's supply chain provides important background information necessary to put the problem this thesis addresses into context. Without understanding the supply chain in detail, there won't be an understanding of why a comprehensive container selection optimization engine is needed. This chapter focuses on the different order types most large, multi-national retailers have, how products are transported from the factory to the DC, and why containers are pooled and repacked at different points in the supply chain. All of these have an impact on what product is inside a container, when it arrives at the DC, and when or if that product is needed.

5.1 Order Type[6]

In supply chain management, all product, regardless of company or industry, can be labeled as either build to order or build to stock. At a large, multi-national retailer, build to order product can be defined as a contract determined quantity of product produced by a retailer for a retail customer where the customer buys a specific quantity of a SKU in advance of a season.

Build to order product can only be placed by retail customers, but large, multi-national retailers typically have internal retail accounts as well. No safety stocks are associated with build to order products because contracts dictate the exact number of products to be created. Once these orders are delivered to the retail customer, consumers can purchase product from them.

At a large, multi-national retailer, build to stock is part of the "Always Available" (AA) channel, and the build to stock is specifically comprised of AA's "at-once" division. AA contains SKUs which are available for sale throughout the year and AA at-once is the AA product which is sold without contract. The AA at-once product is typically sold by a large, multi-national retailer to a customer or a consumer through the internet and is known as the Direct to Consumer (DTC) channel.

Future demand for AA at-once product is not deterministic, so forecasts are used to inform large, multi-national retailers of how much product to order and how much product to keep in reserve at a DC as safety stock. The at-once product is stocked at the DC until purchased by a customer or consumer. Demand forecasts are done in weekly intervals and are aggregated to a product's style-color level.

[6] Gabris, "Size Curve Optimization for Replenishment Products" 15-16

DTC can also place build to order contracts based on forecasted demand for product by consumers. Therefore, product flowing through the DTC channel can be either build to order or build to stock. This product is held at a DC until ordered by a consumer.

It's important to understand how these order types impact a DC's inventory and receiving goals. A DC must hold build to stock and build to order product for the DTC channel, and it must hold product for AA's at-once channel for purchase by retail or wholesale consumers, as well as all other build to order products by retail or wholesale customers. Regardless of the channel that product flows through, the goal of a DC is to receive the product on or before the date it needs to be in that DC so that it can be delivered to the customer on the date promised.

The table below summarizes the different product flows which flow through the DC:

Channel	Build to Order	Build to Stock	Sold to Customer	Sold to Consumer
Futures	x		x	
AA	x		x	
AA At Once		x	x	
DTC	x	x		x

5.2 Network Logistics and Nodes

Regardless of whether a product is build to order or build to stock, it flows through the supply chain in the same manner. At the product source, product is consolidated in containers so that economies of scale in shipping can be seized, and the containers travel between nodes in the supply chain via truck, ship, train, or plane.

The distance a container must travel and its transportation mode are the determining factors of the time the product is in transit. For example, a product made in Asia and transported to a North American DC can take more than four weeks due to the container traveling via truck, ship, and train, and the need to clear customs. Product which is made closer to the DC, in Mexico for example, can be transported by truck and it takes considerably less time to reach the DC.

When a customer orders a product, a sales order (SO) is created in the enterprise management system. The large, multi-national retailer takes the sales order and creates a purchase order (PO) for that product to be made at one of its factories. A PO is also created to satisfy build to stock forecasts. One problem that is encountered by large-multi-national retailers is that the original SO or PO characteristics may change during a product's transit through the supply chain. Examples of changes to the supply picture are a container ship delayed due to a port strike, a container isn't scanned at a certain

location, or problems at customs. A shipment which experiences a delay in the supply may arrive at its DC later than allowable and could be late to the customer.

To prevent this from happening from a supply side perspective, depending on the importance of the product, an exception process may be initiated to expedite the transportation of that product. An example of this exception process is if a factory is late producing an important line of shoes and that product then misses the ship it was intended to be on. That order may be expedited by flying it across the ocean and the order now arrives several weeks earlier than it was originally intended to.

Likewise, changes may occur to the demand picture. Examples of this include a customer cancelling an order, changing the quantity ordered, or going bankrupt. This results in an imbalance which the retailer must try to fix. One way it can fix this problem is shuffling product around - the product from a canceled order can go to a customer which has requested additional product.

Once the product is built at the factory and begins its transit through the supply chain inside a container, changes such as the ones mentioned above occur to the supply and demand properties for each order. The match between the supply and demand characteristics can be thought of as a "supply/demand picture", and that picture is ever-changing based on the variabilities mentioned above. Between the time a container is originally packed and the time it is received at a DC, that supply/demand picture may become sufficiently mismatched, or skewed, that the original characteristics of the product being supplied and demanded do not match anymore. The probability of the supply/demand picture changing for any container increases with the time the container spends in transit and how many products or orders are packed in a container.

The length of time a container is in transit is important because it is directly related to the probability some aspect of the supply or demand picture will change. The number of orders within a container is also important: if there is a probability that an aspect of the supply/demand picture changes for any individual order, that probability can be multiplied across the number of orders within the container to determine the likelihood the container as a whole has a mismatched supply/demand picture. To illustrate what this means, assume there is a 5% chance on average that some aspect of the supply/demand picture changes over a container's trip through the supply chain. If there are 15 orders in that container, the probability that container will have a skewed supply/demand picture is ~54%.

To help combat this problem, there are nodes in the supply chain which provide an opportunity to pool and repackage containers according to the most up-to-date sup-

ply/demand information. These nodes double as locations where changes in transportation mode occur (e.g. a port to move cargo from vessel to train), with the final node being the DC. From the DC, product is shipped to a retail customer.

Below is an illustrative example of the supply chain transporting products from factories to the DC. It shows how nodes are strategic points of pooling as well as where the mode of transportation changes.

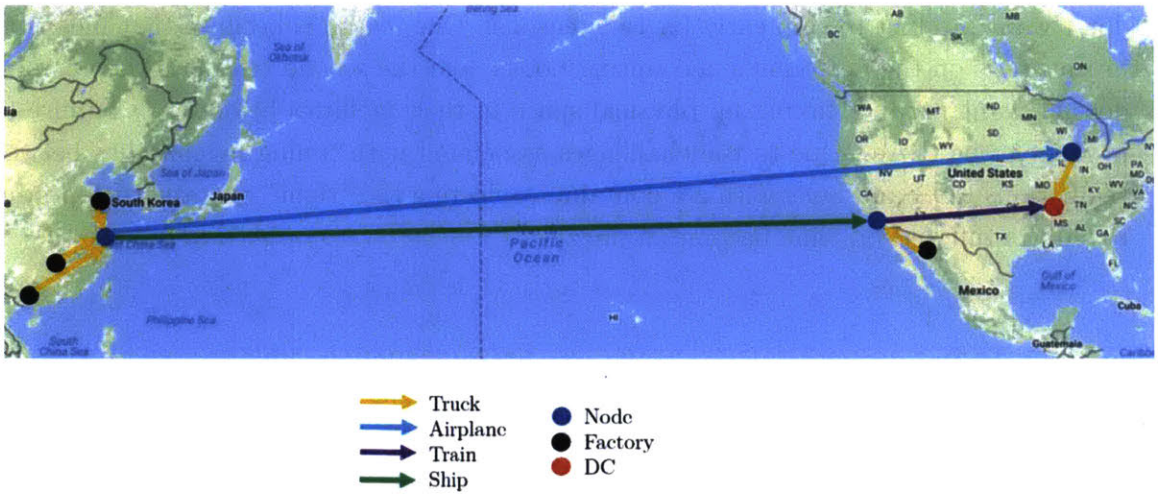

Figure 1 Supply Chain System and Nodes

5.3 Logistical Challenges

As large, multi-national retailers grow in size, more product flows through their supply chains. This means more containers flow through the supply chain and more containers are inbound to each DC.

Customers are expecting increasingly higher levels of service and flexibility through online ordering and speed of delivery, partially due to such offerings as Amazon Prime. Meeting these customer demands means that there is an increased probability that the supply/demand picture becomes skewed over time.

The end result for a large, multi-national retailer is that by the time a container arrives at a DC, there is a high probability that container has mismatched PO and SO items with respect to timing. To illustrate the challenge this creates, assume there is a hypothetical container which arrives at a DC. The container has some product which may be not needed at all because its order has been canceled, some product which is late due to a transportation delay earlier in the supply chain, it wasn't expedited because it is a lower priority level and the product was repackaged into this container at a node, and

still other product which is early because it is high priority, was expedited, and then repackaged into this container at a node.

The challenge of this situation lies in the decision of which containers to receive in which order so that orders can ship out in full and on time. For an order to ship in full and on time, it means that 100% of the quantity ordered for each item within that order and 100% of all items for the order must be located inside the DC and be picked, packed, and shipped on time. On time means a product doesn't ship early or late to a customer.

Customers don't want product to ship late for the obvious reasons, but customers don't want product to ship early for two reasons. One reason is because that means the inventory is on the customer's accounting books, and the second reason is the customer doesn't want product taking up physical space in their facilities before they require the product to be there. Due to the challenges associated with timing mismatches between the various PO's and SO's within a container, selecting the "right" container for product to ship in full and on time becomes a puzzle.

6 Project Motivation & Current State

Now that the reader understands the complexities involved with such a large supply chain, the challenges caused by those complexities can be described. This chapter details the specific problems that have been experienced by large, multi-national retailers and the genesis of this thesis. By reading this section, the reader should be able to understand the author's proposal for a "clean slate" approach to developing a lean inventory management system and optimization model for selecting containers for receipt at a DC.

6.1 Project Motivation

Due to the need to ship outbound product from a DC in full and on time, and the timing mismatch of PO's and SO's, partial orders accumulate within a DC. These partial orders are awaiting additional product to enter the building so an order can be completed, picked, packed, and shipped out.

The challenge is selecting specific containers for receipt in a manner which completes the partial orders inside the building to make these orders shippable. If the containers selected for receipt on any given day do not "unlock" these partial orders, then that product will sit in inventory inside a DC. Another, albeit temporary, way for product to take up inventory space is for product to be brought into a DC too early.

As partial orders and orders which are not ready to ship accumulate inside the building, a bottleneck for new product entering the building is formed. A DC needs empty shelf space in order to receive additional product, complete outbound orders, and ship these orders out. In this way, a large, multi-national retailer's DC's *flows* product through them. Filling a DC with product which doesn't complete orders means the building cannot accept new product and in an extreme case, the building can become gridlocked.

If a DC is filled with product which doesn't complete orders and its outbound shipping rate is restricted, a drop in customer service performance and revenue generation is likely to occur. This happens because a DC cannot flow product through it and there is a drop in the number of shippable outbound orders. As orders ship late because of the drop in ability to ship outbound orders, this creates a drop in on-time customer service performance due to the orders reaching the customer late and also a reduction in revenue generation - large, multi-national retailers are typically paid when product is delivered to the customer.

The number of containers which arrive at large, multi-national retailer's DC's varies throughout the year - forecasted demand and build to order orders ebb and flow through seasons and a container backlog may occur during the busy times of the year. This

backlog builds up when the receiving capacity is less than the incoming supply of containers arriving.

Multi-day backlogs of containers can build up outside DC's for two reasons. One is because only certain containers may be selected to receive product which unlocks partial orders. The second is because large, multi-national retailers must contend with a holiday peak season and the increased amount of demand for product leading up to that peak season. Under the current demand logic of typical large, multi-national retailers, containers with insufficient demand against them are left outside to queue until there is sufficient demand for the product inside. What will be illustrated through this thesis is that there are opportunities in understanding what is truly demanded which will help select the right containers to receive to bring down the storage capacity constraint, and the container backlog.

6.2 Container Receiving Selection Method

The process of receiving containers begins with delivering the containers to a DC's yard. This happens throughout each working day and the DC's yard is the final pooling location of containers before they are received.

There are typically two analysts who play key roles in deciding which containers to select for receipt from the DC's yard and their work is performed looking one day forward in time.

One analyst is located at headquarters and is responsible for managing internal stakeholders' product prioritization requests, which in turn, come from customer accounts. These requests are input into a computer system as a "hot materials list" and represent the demand by customers for certain products. After the prioritization requests are put into a computer system, a container priority list called the "hot list" is returned.

This list is the primary means of identifying important containers to receive and is the main tool used to combat the supply/demand picture mismatches on a daily basis at the DC. However, gaps were identified in the hot list logic during the investigative phase leading to the decision to use an integer optimization program. These gaps are described in the next section.

The other analyst is located at the DC and makes the tactical decisions of selecting the specific containers to be queued for receipt from the yard. Large, multi-national retailers typically have two receiving shifts at a DC and the receiving queue list is created at the end of one shift. This list has the containers which need to be received over the next 24 hours. In general, the hot list is followed from top to bottom, but there are

adjustments that need to be made to the list based on which containers are actually available for receipt, receiving manpower at the DC, and any operational considerations at the DC which may impact receiving operations.

Once the receiving list is created and the second shift starts receiving the containers on the list, the containers which are in the container yard are trucked to one of multiple unloading bays for receiving. By the time the final containers have been received at the end of the two shifts the next day, a new receiving list has been created for the next 24 hours.

6.3 Current Receiving Logic Opportunities

The hot list is the primary tool used to solve the supply/demand picture mismatches at a DC. It does this by queuing containers for receipt based on the newest and best information available. The hot list information is in a spreadsheet listing containers by row in order of priority, and the priority level is based off several cargo manifest characteristics of each container.

The hot list is a simplified heuristic which is created using "IF-THEN" statements (e.g. IF a container has launch product in it, THEN label the container as launch) to help analysts in their task of selecting containers for receipt. Evidence of hot list demand signals being mis-leveraged may be found in the course of a supply chain investigation.

The cargo manifest characteristics come from multiple data sources and the hot list displays these cargo manifest characteristics in the columns along the same row as the container it is attached to. Two examples of the opportunities in the current logic for the hot list are described below, along with an explanation of the ill-effects the logic can cause. While these two can easily be fixed there were other myopic governing logic statements in place which do not view the receipt process as a holistic system.

Logic: Container selection prioritization assumes the supply/demand picture has not changed significantly and is therefore agnostic to the inventory in the DC and the outbound demand.

Effect: This results in receiving a specific product even when there is sufficient product inside to meet outbound demand because one must err on the safe side of ensuring product from the "hot materials list" is received to prevent stockouts; this is especially true with build to stock product. If the quantity of product inside the DC exceeds the required safety stock levels but the product is still prioritized for receipt, it will sit in inventory instead of flowing through the DC.

Example: Based on this logic, the number of AA at-once products inside a DC was found to exceed the safety stock requirements in a space constrained facility.

Logic: Each container on the hot list is flagged and prioritized by the highest brand-valued product inside regardless of when the product is needed.

Effect: Consider a hypothetical situation of two containers available for receipt at a DC, one container ranked as very high priority but required one month away, and the other labeled as medium priority but required in two days. Given the ability to receive one container, the stated logic would select the first container at the opportunity cost of the second. In addition, because the entire container is labeled as the most important product inside it, if the second container had low priority product inside but that low priority product would complete several outbound orders for customers, receiving the first container is an even higher opportunity cost.

Example: For each container on the hot list, a column shows the percentage demand of each container against the original SO-PO match (i.e. when the product was initially intended to arrive at the DC to ship out on time) in monthly buckets. There are instances where the following example situation occurred. Say there are 100 medium priority orders in a container and there is a demand during the current month for 80% of those orders and another container with 99 low priority orders and 1 high priority order with a demand for the current month for 30% of those 100 orders, but the high priority item is not needed until the following month. If only one container can be received, the logic states to receive the second container because there is a high priority item in it, even though there is less demand against the container and the high priority product is not due for another month.

Because the frequency for potential problems such as these exist at large, multi-national retailers and it can be difficult to determine the exact extent to which the hot list logic could be an accurate heuristic for container selection, a "clean slate approach" for designing the receiving process was the chosen course of action.

By not viewing the outbound demand for product, product in inventory, and the entire cargo manifest in each container at once, the focus of what to receive at a DC becomes centered around the supply characteristics of containers. By focusing on the supply, analysts are given incomplete information and asked to solve a very difficult puzzle. This can be described is a push supply chain through the DC, pushing product in rather than pulling it in based on outbound demand and inventory inside.

While a push supply chain makes sense upstream of a DC due to the long lead times associated with sourcing and transporting product overseas, a DC is an ideal location to

implement a pull system. The inventory inside a DC is used as a buffer against demand in the near future, while the containers can be received to make up the system deficit for the remaining demand. In order to turn the container selection process from a push supply chain into a pull supply chain, a lean inventory management system needs to be constructed as well as an optimization engine which calculates the best containers to receive based on the inventory management system's outputs.

7 Lean Inventory Model Formulation

The lean inventory model presented in this section has one purpose: to generate an important parameter for the optimization engine presented in Chapter 12. The lean inventory model generates a list of all SKU's p and the quantity of each SKU p which is demanded. This list is what should be targeted for receipt today. The lean inventory model is simple arithmetic and no algorithms are involved. Therefore, the model described in Section 11 is pre-work performed to generate an input to the optimization model.

The lean inventory and optimization model should be run each morning to generate the day's receipt plan. The receipt plan will change each day based on the containers which arrive at the DC's yard and changes in the inventory data and demand data. By running the program each morning, the information used in the model is timely and enables the analysts to select the best containers for receipt.

The first step in developing the lean inventory model is to aggregate all outbound demand for product within a defined future time window. The process of selecting a future time window is in itself an optimization because too long of a time horizon will include too many sales orders and potentially de-prioritize containers which are needed imminently. Conversely, too short of a future time window may result in important product shipping late because it wasn't received in time, and this is unacceptable.

The size of the future time window is dependent on the size of the facility and the degree of confidence that the supply/demand picture won't change. The smaller the facility, the shorter the time window should be so that the facility doesn't become clogged with product, whereas a larger facility can use a longer looking horizon. The more confidence there is in the supply/demand picture remaining the same, the longer the time window is allowed to be, because there is confidence that once the product goes into inventory the demand against it won't change. Likewise, if the supply/demand picture is constantly changing, making receipt decisions too far in advance could result in sub-optimal results because the information becomes outdated after the container is received.

An appropriate range should be forward looking between one and three weeks. Because this is a general formulation, a specific time window size will not be specified here.

The aggregation of demand for the lean inventory model needs to be done for both build to order and build to stock product. Build to order involve set contracts and there is a higher degree of confidence in a contract's demand materializing, so the time window can be relatively longer for build to order than build to stock. Build to stock relies on

forecasts and because one of the central tenets of forecasts is that they are more accurate in the near-term range than long-term range, the forecast time window is kept as small as reasonable. In this thesis, the demand aggregation of build to order and build to stock will be done separately and then combined in the end. The total demand can then be compared to the inventory in the DC to determine how much of what product must be received to meet all orders within the time window.

7.1 Build to Order Demand Signal Creation

The build to order demand signal creation is relatively straightforward. The goal is to sum up the total demand for each individual product across all customers within the chosen time window. The result of this shall be called the "raw build to order demand signal"

Time Periods

It is assumed the number of days required for processing inbound and outbound product within the DC is a known constant. This constant is subtracted from the outbound shipping date to arrive at a required receipt date. The required receipt date for any product is represented by the letter "t".

For example, if it takes one day to receive product, and the pick pool size is kept at two days worth of work, and it takes one day for that product to ship out, then a product needs to be selected for receipt no less than four days prior to the day it needs to ship out to the customer. If a customer demands three hundred units of product on the 16^{th} of a month, that product must be received by the 12^{th}. This required receipt date calculation is pre-work for all outbound demand before the inputs go into the optimization model.

Stock Keeping Units

A stock keeping unit (SKU) is the unique product-color-size identifier for a product which is ordered by a customer or consumer. The style for a product is represented by the letter "j", the color by the letter "k", and size by the letter "l". When putting the style-color-size together to form a SKU, it can be represented by the letters "jkl" or simply the letter "p".

Customer Identifier

Each individual customer needs to be identified because the customer needs their product to be received by the DC on a certain date for it to ship on time. Each customer has a unique identifier represented by the letter "i".

Notation Definitions

d_p= total quantity demanded for SKU p, or raw build to order demand signal, in units
q_{pti} = quantity of SKU p demanded on day t by customer i, in units

Aggregating all build to order product can be described mathematically as:

$$d_p = \sum_{i=1}^{I} \sum_{t=1}^{T} q_{pti} \tag{1}$$

This formulation sums the build to order demand but does not capture the build to stock demand as build to stock demand is forecasted and therefore must be handled differently.

7.2 Build to Stock Demand Signal Creation

Build to stock orders at large, multi-national retailers are forecasted in weekly buckets. Forecast calculations are performed to ensure the quantity of product in inventory in a DC at the beginning of the week sufficiently covers all demand during that week. A DC's goal is for the quantity on hand of product at the end of a week to equal the safety stock at the end of that week. Safety stock calculations are not in the scope of this thesis, it is assumed that they are provided in the forecast.

Because forecasts are intended to forecast all demand for a specific week's product, they need to be adjusted as product sales for that week occur. Put another way, as sales for build to stock product materialize prior to the week's beginning, the forecast needs to be reduced or else there would be double counting of demand. If the sales quantity for a product is equal to or greater than the forecasted quantity for that product, the forecasted demand would be zero for that product - there are no negative forecasts. Because these sales occur before the week being forecasted but are intended to ship to the customer during the forecasted week, it is simply a re-allocation of forecasted demand to materialized demand by one or more customers.

For example, if 300 units are forecasted for week X, and 200 units are sold for week X during the week before week X, the forecast needs to be reduced because this demand for 200 units has already been accounted for in the forecast. By updating the forecast, the original 300 units of forecasted demand drops to 100 of forecasted demand, while 200 units become materialized demand. Decrementing the forecast like this results in an "unconsumed forecast" of 100 units. The remaining forecast of 100 units is the demand signal for product to be brought into the DC; it's forecasted that 100 units will be ordered during the week. If 300 units are forecasted for week X, and 400 units are sold for that week, the unconsumed forecast would be 0 units.

Forecasted Demand & Unconsumed Forecast

The forecast is calculated at the style-color level which is one level of granularity larger than the SKU level (style-color-size). The style for a forecasted product is represented by the letter "j", and the color by the letter "k". The unconsumed forecast is calculated to the SKU level, and adds the size represented by the letter "l".

Materialized Sales

As sales for build to stock product occur for a specific week, as in the example above, those materialized sales need to be summed up and subtracted from the forecasted demand. Because the forecast is at the style-color granularity, and materialized sales occur at the style-color-size granularity as well as for specific customers, the at-once sales need to be aggregated to the style-color level. This is done by summing across all sizes and across all customers to match the forecast granularity. The style for a product is represented by the letter "j", the color by the letter "k", size by the letter "l", and customer by the letter "i".

Notation Definitions

u_{jk} = quantity of unconsumed forecast for style j and color k, in units

f_{jk}= quantity of initial forecast for style j and color k, in units

m_{jkli}= quantity of materialized sales for style j, color k, size l, and customer i, in units

The unconsumed forecast can be described mathematically as:

$$u_{jk} = \begin{cases} f_{jk} - \sum_{i=1}^{I} \sum_{l=1}^{L} m_{jkli} & if \quad f_{jk} - \sum_{i=1}^{I} \sum_{l=1}^{L} m_{jkli} > 0 \\ 0 & if \quad f_{jk} - \sum_{i=1}^{I} \sum_{l=1}^{L} m_{jkli} \leq 0 \end{cases} \tag{2}$$

This unconsumed forecast for product now needs to be disaggregated into the style-color-size for the purposes of determining what to receive. To do this, size curves can be applied. A size curve is a projected breakdown of the sizes demanded by consumers for a product. The demand for each size is expressed as a percentage of the total demand. For example, if a t shirt has sizes small, medium, and large, a size curve may be 30% small, 50% medium, and 20% large. Applying the size curve at this stage is necessary to avoid the mistake of "satisfying" forecasted demand at the style-color level by bringing in only one size of product.

When the size curve is applied to each product at the style-color level, it becomes a SKU.

Size Curve Multiplier

The size curve multiplier is the percentage which the total number of style-color products needs to be multiplied by in order to determine the unconsumed forecast at the SKU level. As before, the style is represented by the letter "j", color by letter "k", size by letter "l", and a SKU is represented by the letters "jkl".

Notation Definitions
a_{jkl} = quantity of unconsumed forecast for SKU jkl, in units
e_{jkl} = percentage size curve multiplier for style j, color k, and size l, in %
u_{jk} = quantity of unconsumed forecast for style j, and color k, in units

Applying the size curve to the unconsumed forecast can be described mathematically as:

$$a_{jkl} = e_{jkl} u_{jk} \tag{3}$$

The unconsumed forecast only captures the estimated demand for the week, and does not include the number of units required in inventory as a safety stock to act as a buffer against uncertainty in the forecast. Because the forecasted quantity is expected to be consumed during the week ending at the safety stock level, there must be sufficient inventory in the safety stock to cover higher than forecasted demand and avoid stock outs. Therefore, this safety stock must be included in the lean inventory model.

The safety stock is calculated in the style-color level, so the same size curve used above needs to be used on the safety stock level.

Notation Definitions
h_{jkl} = quantity of safety stock required for SKU p, in units
e_{jkl} = percentage size curve multiplier for style j, color k, and size l, in %
z_{jk} = quantity of safety stock required for style j and color k, in units

The calculation of the safety stock required for each SKU in the forecast can be described mathematically as follows:

$$h_{jkl} = e_{jkl} z_{jk} \tag{4}$$

The various components of the build to order forecast need to be added together to determine the total quantity of each SKU required to be on hand at the beginning of a week. This will be known as the "raw at-once demand signal." The components of this are the unconsumed forecast at the SKU level, the materialized sales due for receipt at the DC during the week, and the safety stock level at the SKU level.

SKU Identifier

Now that all products have been disaggregated into the SKU's, a new letter, "p" is used to represent the SKU. The letters "p" and "jkl" are equivalent.

Notation Definitions

s_p = raw demand signal for build to order and build to stock for SKU p, in units

h_p = quantity of safety stock required for SKU p, in units

a_p = quantity of unconsumed forecast for SKU p, in units

m_{pi}= quantity of materialized sales for SKU p and customer i, in units

The raw at-once demand signal for the build to stock material can be calculated as follows:

$$s_p = a_p + h_p + \sum_{i=1}^{I} m_{pi} \tag{5}$$

Now that the raw demand signals for build to order and build to stock have been calculated, the inventory inside the DC can be compared against these demand signals to calculate what must be received to make up the remaining demand. If there is an inventory quantity equal to or greater than the quantity demanded, that specific SKU should not be considered "in demand." If a SKU is not in demand it should not factor into the decision to receive a container.

Likewise, if the inventory of a SKU inside the DC is less than the quantity demanded for that SKU, the deficit should be received in order to complete orders. The deficit shall be called the "refined demand signal" and the optimization engine will only take the refined demand signal into consideration when selecting containers for receipt.

Notation Definitions

r_p = refined demand signal for SKU p, in units

x_p = quantity of SKU p in inventory at a DC, in units

s_p = raw at-once demand signal for SKU p, in units

d_p = raw build to order demand signal for SKU p, in units

Calculating the refined demand signal mathematically can be done with the piecewise function:

$$r_p = \begin{cases} 0, & if \quad x_p - s_p - d_p \geq 0 \\ s_p + d_p - x_p & if \quad x_p - s_p - d_p < 0 \end{cases} \tag{6}$$

The total quantity of required SKU's for each SKU over the build to order time window and the build to stock time window is captured in this r_p.

7.3 Model Assumptions

There are assumptions made in the formulation of this model

Mathematics of Size Curve Applications

The calculation proposed for applying the size curve to both the forecasted demand and the safety stocks, equations (3) and (4), takes the size curve and multiplies it by each product's style-color to arrive at a style-color-size. This is a simplified formulation of performing a size curve disaggregation resulting in "lower-than-anticipated service levels for the size-level stock keeping units, since the style/color safety stock does not account for the increased forecast error at the size level...this additional size level error can be accounted for by right-sizing safety stock." [7] For further information on setting a size curve and setting a safety stock at the size level, see "Size Curve Optimization For Replenishment Products" by Andrew Gabris, LGO '16.

The assumption is that one can simply disaggregate a product's style-color into style-color size using the size curve. This is an assumption because the accuracy of a forecast is higher when product is aggregated. There's variance around forecasted demand for a product, and there's variance around the forecasted size breakdown of a product. For example, given two people of different builds but otherwise identical including fashion tastes, the assumption states the two would buy the same style-color shirt regardless of their build.

While there is a correlation with the color chosen by two people of different builds but otherwise identical taste, it cannot be assumed to be perfectly correlated. Therefore, to gain a higher degree of accuracy, one must account for the variance of both the forecasted demand for a product's style-color as well as forecast for its sizing before disaggregating the style-color into style-color-size. For the purposes of this model, this has been ignored.

Build to Order Assumptions

There are several methods to replenish build to order stock. There are applications where some are better suited than others. For example, a simple regularly scheduled replenishment model where one orders what has been consumed between review periods is useful for products which have a relatively consistent demand, a forecast model is

[7] Andrew Gabris, "Size Curve Optimization for Replenishment Products"

better suited for products which have a higher standard deviation in their demand, and a Newsvendor model is useful for seasonal orders with a forecasted total demand for the season.

The assumption used in this model is that the company uses a forecast model where a forecast for the total demand in a week and safety stock levels are calculated. The generation of this forecast and its implications on sourcing and lead times are out of the scope of this thesis. The model is only concerned with receiving sufficient supply of SKU's in the DC yard to meet the forecasted demand and safety stock levels.

8 Optimization Engine Formulation

The only term carried over from Chapter 11 is the final term- the refined demand signal r_p. This represents the quantity of SKU p demanded. The goal of the optimization engine is to select the best combination of containers to satisfy the refined demand signal. This is an important step towards completing an order and making that order shippable, but it doesn't necessarily mean any single order can be completed (all line items are filled) and shipped out. The focus of the engine is on completing the individual line items within an order. Because there is a probability that any container which is selected by the engine for receipt will contain the final quantity of product to fill a line and complete an order, i.e. it is not random, this approach employs the law of large numbers. It relies on the assumption that over time, a maximization of the demand satisfaction for r_p will complete all the line items for an order and that order will ship. Each objective function will select containers based on filling r_p for line items within orders and employ the law of large numbers to make orders shippable.

Because there are instances where certain SKU's may be prioritized over other SKU's, there needs to be multiple objective functions. The computer can only run the optimization program on one objective function at a time meaning the objective functions are separate optimization engines. For the purposes of this thesis and ease of understanding the goals of the optimization the objective functions will be listed together in this section.

There are four objective functions in total: maximize throughput, maximize specific demanded segments, maximize revenue, and maximize a balance of the first three. The objective functions will focus on finding the best set of containers to receive based on the needs of the user. Given the same inputs, each engine may select different sets of containers to receive because the goals of each objective function differ from each other.

8.1 Objective Functions

The first objective function, maximize throughput, selects the set of containers which maximizes the unit demand satisfaction of r_p. By maximizing this unit demand satisfaction, the model works to complete line items in orders.

Objective function two, maximize specific demanded segments, adds a 1-10 prioritization scalar to each SKU p for user prioritization. For example, if a certain shoe were demanded by a retailer, there needs to be a way to artificially ensure that product is prioritized for receipt. This is different from the current hot list because if that demanded shoe is not actually in the refined demand signal then it is ignored. This ensures the artificial prioritization is not accidentally tied to product which is not needed.

The third objective function, maximize revenue, uses r_p and a 1-10 prioritization scalar proportional to the revenue generated from any SKU p to select containers which maximize the revenue of product r_p received. This objective function could be used for a period of time towards the end of a quarter, for example, to help identify which containers would complete as many line items as possible for high revenue product.

The final objective function, a maximization of a balance between the first three, uses the refined demand signal r_p, a 1-10 prioritization scalar, and a 1-10 revenue scalar to select containers which satisfies a combination of the above.

Regardless of which objective function is used, it is a maximization of fulfilling the refined demand signal and, if applicable, any prioritization scalars which may be desired by the business to prioritize objectives other than just the maximization of the refined demand signal.

In order to select the containers sitting in the yard which have demanded SKU's in them, the list of SKU's in the containers must be compared against the list of SKU's which are in demand. The intersection of these two lists will be dubbed "matched product."

Matched Product

Matched product is defined in this thesis as the intersection of SKU's which are required in the refined demand signal and SKU's which are contained within the container supply in the DC yard ready for receipt. For example, if one or more red medium t shirts are demanded and one or more red medium t shirts is in one or more containers, then it's a matched product. Similarly, if that red medium t shirt is not located in a container, or if that red medium t shirt is not demanded but located in a container in the DC yard, it is not a matched product. Matching demanded product to the supplied product in the container is done at the SKU level with the letter "p".

Notation Definitions

R = Set R is all SKU's p within refined demand signal
Q = Set Q is all SKU's p within container supply in DC yard
Y = Set Y is all SKU's p within both refined demand signal & container supply in DC yard

This can be expressed mathematically as:

$$Y = R \cap Q \tag{7}$$

It is shown visually in a Venn diagram as:

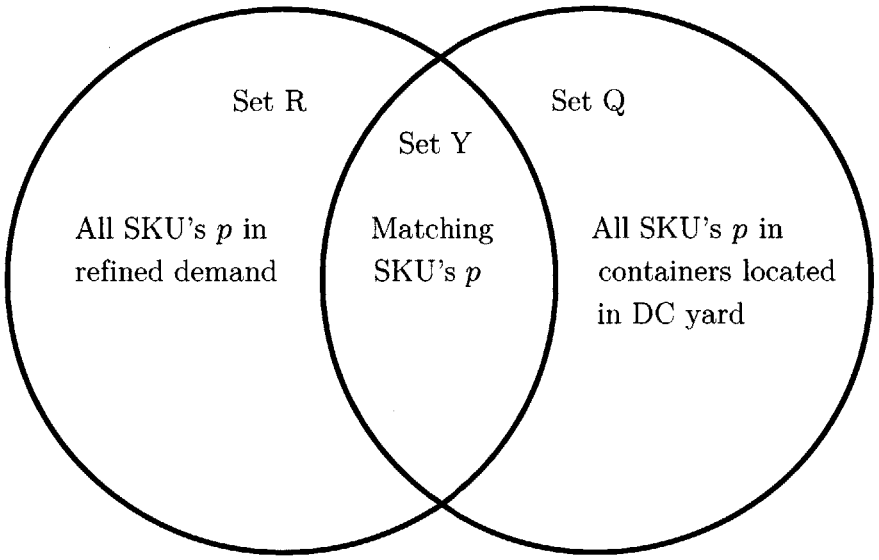

Figure 2 Calculation of Matched Product y_p

Objective Function One: Maximize DC Throughput

The purpose of this objective function is to maximize the quantity of demanded products brought into the building. It maximizes the satisfaction of the refined demand signal.

Notation Definitions

y_p = quantity of matched SKU p, in units

The throughput objective function is:

$$\max \sum_{p=1}^{P} y_p \tag{8}$$

Objective Function Two: Maximize Specific Demanded Segments

This objective function maximizes the sum product of a segment priority and the matched product to ensure the most important products are received. A segment priority is a number to artificially prioritize certain matched SKU's for receipt. The purpose

of this is to add the product priorities used in the hot list, but do it with finer granularity than what is currently done.

Its use and advantage over the hot list is illustrated with a simple example: assume there is one container with 75 high priority and 25 low priority products inside and another container with 100 high priority products inside. If the constraint is that only one container can be received, the business needs a way to select the high priority container. By assigning a prioritization scalar to each high priority product, the engine will select the container with 100 high priority products. In the current hot list, both containers will appear to be equally important because they'd both be labeled as high priority.

Segment Priority

In order for a segment (e.g. all SKU's which are launch product, DTC, running, football, etc...) to be properly prioritized, each segment must have a scalar assigned to it by the user before the optimization engine runs. This scalar is used in the objective function to force the program to adjust the decision variable by prioritizing certain products. This segment priority scalar is defined for each unique SKU as pre-work before the optimization engine runs.

Notation Definitions

α_p = 1-10 scalar prioritization for segment p

The segment maximization objective function is:

$$\max \sum_{p=1}^{P} \alpha_p y_p \tag{9}$$

Objective Function Three: Maximize Revenue

This objective function maximizes the revenue generated by selecting for receipt the highest value product which is needed to meet demand.

Revenue Scalar

The revenue scalar is created by making a 1-10 scalar directly proportional to the range between the lowest and highest revenue generating SKU's, respectively. This is done so that in objective function four, the two scalars, α_p and β_p are equally weighted and the revenue scalar doesn't "overpower" the segment prioritization scalar. To accomplish this, the revenue scalar needs to be calculated for each SKU before going into the optimization engine. The calculation is done as follows:

$$\beta_p = 9/(\$_\mu - \$_\pi) * (\$_p - \$_\pi) + 1 \tag{10}$$

where:
β_p = 1-10 scalar for revenue generation of SKU p
$\$_\mu$ = revenue generated by highest revenue SKU μ, in dollars
$\$_\pi$ = revenue generated by lowest revenue SKU π, in dollars
$\$_P$ = revenue generated by SKU p, in dollars

For example, if there are 6 SKU's, and they generate \$5, \$6, \$10, \$15, \$22, and \$30 each, the scalars would be 1, 1.36, 2.8, 4.6, 7.12, and 10, respectively.

Notation Definitions
β_p = 1-10 scalar for revenue generation for SKU p

The revenue maximization objective function is:

$$\max \sum_{p=1}^{P} \beta_p \, y_p \tag{11}$$

Objective Function Four: Maximize Balance

This objective function maximizes a balance between the three other objective functions. Its formulation is comprised of both the segment priority scalar and the revenue scalar. These two scalars have equal impact on the objective function, and the algorithm changes the selection of containers to maximize the quantity of matched products within the containers multiplied by the two scalars. This objective function's formulation is:

$$\max \sum_{p=1}^{P} \alpha_p \beta_p y_p \tag{12}$$

8.2 Decision Variables

The optimization engine works to maximize each objective function by changing the quantity of matched product y_p in the objective function by selecting certain sets of containers. The decision to select or change y_p and therefore whether a container is selected or not are both decision variables.

The cargo manifest of each container in the DC yard is known, which is to say, the quantity of each SKU in each container is known. When a container is in the DC yard and available for receipt, the original PO tied to each SKU can be ignored; it does not help answer which containers should be received to best fulfill the refined demand signal.

When a container is received, the entire container is unloaded and all product inside goes into inventory. The decision to receive a container is binary and the selection of a container to receive is the only decision the program can make to impact the objective function. The letter "b" represents the decision to receive or not receive, and the letter "c" represents a container. The optimization program is an integer program because the decision variable is constrained to be binary.

Containers located in the DC yard are uniquely identified and are represented by the letter "c". The decision variable is represented by the letter "b" and it equals 0 if container "c" is not selected for receipt, and equals 1 if the container is selected for receipt.

8.3 Constraints

The constraints on the program serve to provide bounds on which containers are selected in order to maximize the objective function.

Binary Decision Variable Constraint

Containers located in the DC yard are uniquely identified and are represented by the letter "c". The decision variable is represented by the letter "b" and it equals 0 if container "c" is not selected for receipt, and equals 1 if the container is selected for receipt.

Notation Definitions

b_c = container c is selected for receipt or not

The decision variable is then constrained as follows:

$$b_c \in \{0,1\} \tag{13}$$

Receiving Capacity Constraint

The DC can only receive a certain number of units per day. The number of units it can receive is based on the number of shifts and rate at which units within the containers are received. This receiving capacity may change each day and the total quantity of product received cannot exceed that capacity. The maximum number of units which can be received in a day is represented by the letter "k".

Notation Definitions

q_{cp} = quantity of product p within container c, in units

k = maximum number of units which can be received by the DC in a day, in units

The receiving capacity constraint is:

$$\sum_{c=1}^{C} \sum_{p=1}^{P} b_c q_{pc} \leq k \tag{14}$$

Quantity of Matched SKU's Cannot Exceed Quantity of Received SKU's Constraint

Each objective function selects containers to maximize the the sum product of matched SKU's and (if applicable) prioritization and/or revenue scalars. The total quantity of each SKU that can be maximized through the objective function must be constrained to not exceed the total quantity of that SKU within the containers selected for receipt. This constraint is formulated as:

$$y_p \leq \sum_{c=1}^{C} b_c q_{pc} \ \forall p \in P \tag{15}$$

Collatoral Product Prevention Constraint

The quantity of product which impacts the maximization of the objective function cannot be allowed to exceed the quantity of product that is required by the refined demand signal. For example, if 120 units of a blue large t shirt were the refined demand signal and 200 units were received, the objective function should only be increased to a maximum of 120. The additional 80 units of product which were brought in can be considered "collateral" product which was only received because when a container is selected for receipt, the entire container must be received. This constraint is formulated as:

$$y_p \leq r_p \ \forall p \in P \tag{16}$$

Non-Negativity Constraint

The final constraint imposed on the program is requiring the number of matched products to be positive. This constraint is formulated as:

$$y_p \geq 0 \ \forall p \in P \tag{17}$$

8.4 Model Formulation

With the description of the model complete, the complete formulation of the model is displayed below:

Parameters and Definitions

α_p = 1-10 scalar prioritization for segment p

β_p = 1-10 scalar for revenue generation of SKU p

q_{cp} = quantity of product p within container c, in units
k = maximum number of units which can be received by the DC in a day, in units
r_p = quantity of SKU p in refined demand signal, in units
b_c = container c is selected for receipt or not

Decision Variable

b_c = container c is selected for receipt or not
y_p = quantity of matched SKU p located in intersection of refined demand signal and containers located in yard ready for receipt, in units

Constraints

Binary Decision Variable Constraint:
$b_c \in \{0,1\}$

Quantity of Matched SKU's Cannot Exceed Quantity of Received SKU's Constraint:

$$y_p \leq \sum_{c=1}^{C} b_c q_{pc} \quad \forall p \in P$$

Receiving Capacity Constraint:

$$\sum_{c=1}^{C} \sum_{p=1}^{P} b_c q_{pc} \leq k$$

Collateral Product Prevention Constraint:
$y_p \leq r_p \ \forall p \in P$

Non-Negativity Constraint:
$y_p \geq 0 \ \forall p \in P$

Objective Functions

Maximize DC Throughput:

$$\max \sum_{p=1}^{P} y_p$$

Maximize Specific Demanded Segments:

$$\max \sum_{p=1}^{P} \alpha_p y_p$$

Maximize Revenue:

$$\max \sum_{p=1}^{P} \beta_p \, y_p$$

Maximize Balance:

$$\max \sum_{p=1}^{P} \alpha_p \beta_p y_p$$

9 Model Testing

The data for the testing of this model is not real data. It was created to be representative of conditions which could occur at a DC, but all the data created was done using Microsoft Excel's random number generator. Several measures were taken to create a hypothetically realistic scenario which demonstrates the efficacy of the model over a heuristic. Both the optimization model results and results from a heuristic are displayed below, and the input data for the model and heuristic is located in Appendix 1.

The measures taken to highlight the complexity of real conditions are summarized below:

1. There are a total of 20 containers available for receipt with a total quantity of 36,946 product units within the containers. Between 1,074 and 2,593 units are contained in each container.
2. There are 75 SKU's which have a refined demand signal, randomly assigned a value of 50 to 600. The sum of the required SKU's is 23,264.
3. Each container has a randomly selected number of collateral products, represented by "ABC-111-L", in it. The sum of collateral products in each container is a random variable between 100 and 500.
4. Each SKU p is located in exactly 4 random containers, and in a random quantity between 10 and 200. This ensures that a varying quantity of SKU p is located in any container, products are located in multiple containers giving reason to select one container over another, and each container has a varying number and quantity of products in it.
5. There are three discrete segment prioritization scalar values equal to 1, 5, and 10 assigned randomly to each SKU p.
6. There are ten values for the revenue scalar ranging from 1-10 assigned randomly to each SKU p.
7. The model was tested using all four objective functions against fifteen receiving capacity constraints from 10,000-24,000 units in 1,000 unit increments. This is 60 runs of the model and it took the model less than a minute to run.

The heuristic is calculated in a similar method as each objective function, but is focused at the container level rather than across all containers. It is easy enough to generate a set Y of the "matched products" as it's the intersection of sets R and Q, this is from equation (7). Selecting the right number of matched product y_p to ensure it is less than

what is required r_p is too difficult to perform because there are too many parameters and combinations to consider. Therefore, a myopic answer will be determined by developing a container "score" where the quantity of all matched products in set Y are assumed to be in demand. It's myopic because it ignores the "Quantity of Matched SKU's Cannot Exceed Quantity of Received SKU's Constraint" and the "Collateral Product Prevention Constraint." A sum product is performed for each container where the quantity of each matched product in each container is multiplied by the prioritization scalar and the revenue scalar, then all these products are summed together. Below is the container score calculation for container A:

					Heuristic Container Score Calculator			
Container	SKU	q	α_p	β_p	Throughput	Segment	Revenue	Balanced
A	VKA-848-L	182	2	7	182	364	1274	2548
A	GYV-716-S	196	2	7	196	392	1372	2744
A	ICH-875-S	172	2	6	172	344	1032	2064
A	MCZ-559-M	162	2	2	162	324	324	648
A	BAM-368-L	70	2	2	70	140	140	280
A	LDM-027-M	17	2	4	17	34	68	136
A	FRC-734-M	160	3	1	160	480	160	480
A	SRP-593-M	20	3	2	20	60	40	120
A	LIV-417-M	19	1	3	19	19	57	57
A	ABC-111-L	349	0	0	0	0	0	0
			Sums		998	1793	4467	9077

Once the score for each container is calculated, the list of containers is then ranked from highest to lowest score, and container selections would occur in the order from highest score to lowest score, up to the maximum receiving capacity for that day. Performing the heuristic in this manner allows the user to mimic two of the objective functions, the "Binary Decision Variable Constraint" and the "Receiving Capacity Constraint." The heuristic container scores are listed in Appendix 2.

In the optimization results listed below the "Capacity" is the receiving capacity constraint, the "Number Of Units Revieved" is the total number of units received, "Number of Containers Selected" is the sum of y_p for each capacity constraint, "O.F. Value" is the objective function value, or numerical value that is being maximized in each run, and "Containers Selected" is the specific containers chosen for receipt. For the Balanced and Revenue maximization objective functions, there is an additional column called "Dollars of Matched Product" which is the sum product of the matched SKU's y_p and the revenue scalar β_p.

In the heuristic results below the heuristic score is listed along with a "Calculated O.F. Value" which is a calculated objective function value at the capacity constraint listed. This is done by taking the containers selected using the heuristic and then applying the collatoral product prevention constraint to it. This constraint allows the value of y_p to be correctly calculated and therefore the objective function can be calculated.

Throughput Objective Function

Optimization Results

Capacity	Number of Units Received	Number of Containers Selected	Number of Matched Units	O.F. Value	Containers Selected
10000	9971	6	8228	8228	E,I,K,O,S,T
11000	11000	6	9037	9037	I,K,J,M,O,S
12000	11957	7	9653	9653	I,K,M,L,O,S,T
13000	12752	7	10317	10317	I,K,J,M,O,S,T
14000	13917	7	11020	11020	F,I,K,J,M,O,T
15000	14961	8	11634	11634	E,F,I,K,J,P,S,T
16000	15930	8	12275	12275	E,F,I,K,J,O,Q,T
17000	16802	8	12844	12844	F,I,J,O,Q,P,S,T
18000	17876	9	13548	13548	E,F,I,J,O,Q,P,S,T
19000	18857	10	14043	14043	E,F,I,H,J,L,Q,P,S,T
20000	19828	10	14669	14669	E,F,I,K,J,O,Q,P,S,T
21000	20802	11	15064	15064	B,E,F,I,H,J,Q,P,S,R,T
22000	21977	12	15691	15691	C,B,E,F,I,J,L,O,Q,P,S,T
23000	22571	12	16111	16111	B,E,F,I,K,J,L,O,Q,P,S,T
24000	23929	13	16690	16690	C,B,E,F,I,K,J,L,O,Q,P,S,T

Heuristic Results

Capacity	Number of Units Received	Number of Containers Selected	Heuristic Score	Calculated O.F. Value	Containers Selected
10000	7414	3	7155	6129	F,D,P
11000	10007	4	9319	7701	F,D,P,Q
12000	10007	4	9319	7701	F,D,P,Q
13000	12281	5	11472	9454	F,D,P,Q,O
14000	12281	5	11472	9454	F,D,P,Q,O
15000	14608	6	13331	10659	F,D,P,Q,O,N
16000	14608	6	13331	10659	F,D,P,Q,O,N
17000	16457	7	15180	11632	F,D,P,Q,O,N,H
18000	16457	7	15180	11632	F,D,P,Q,O,N,H
19000	18409	8	16954	12638	F,D,P,Q,O,N,H,K

20000	18409	8	16954	12638	F,D,P,Q,O,N,H,K
21000	20423	9	18718	13487	F,D,P,Q,O,N,H,K,R
22000	20423	9	18718	13487	F,D,P,Q,O,N,H,K,R
23000	22624	10	20433	14936	F,D,P,Q,O,N,H,K,R,J
24000	22624	10	20433	14936	F,D,P,Q,O,N,H,K,R,J

Revenue Objective Function

Optimization Results

Capacity	Number of Units Received	Number of Containers Selected	Number of Matched Units	Dollars of Matched Units	O.F. Value	Containers Selected
10000	9921	6	7686	47502	47502	E,I,H,N,S,T
11000	10861	6	8352	51715	51715	I,H,N,S,R,T
12000	11935	7	9146	56025	56025	E,I,H,N,S,R,T
13000	12976	7	9928	58624	58624	E,D,F,I,H,J,S
14000	13989	8	10526	62661	62661	E,G,I,H,N,Q,S,T
15000	14970	8	11497	66630	66630	E,F,I,H,J,Q,S,T
16000	15981	8	11725	70183	70183	D,F,I,H,J,N,S,T
17000	16984	9	12781	73914	73914	E,F,I,H,J,Q,S,R,T
18000	17995	9	12863	76633	76633	D,F,I,H,J,N,S,R,T
19000	18803	10	13318	79032	79032	E,D,G,I,H,J,N,Q,S,R
20000	19910	10	13827	81543	81543	E,D,F,I,H,J,N,Q,S,R
21000	20786	11	14576	84707	84707	E,G,F,I,H,J,N,Q,S,R,T
22000	21792	11	15388	86659	86659	E,F,I,H,J,N,Q,P,S,R,T
23000	22791	12	15355	88278	88278	E,D,G,F,I,H,J,L,N,Q,S,R
24000	23866	12	15990	90685	90685	E,D,G,F,I,H,J,N,Q,P,S,R

Heuristic Results

Capacity	Number of Units Received	Number of Containers Selected	Heuristic Score	Calculated O.F. Value	Containers Selected
10000	9853	4	49229	43501	F,N,D,Q
11000	9853	4	49229	43501	F,N,D,Q
12000	9853	4	49229	43501	F,N,D,Q
13000	12127	5	59617	49748	F,N,D,Q,O
14000	13879	6	69902	56681	F,N,D,Q,O,T
15000	13879	6	69902	56681	F,N,D,Q,O,T
16000	15893	7	80155	62653	F,N,D,Q,O,T,R
17000	15893	7	80155	62653	F,N,D,Q,O,T,R

18000	15893	7	80155	62653	F,N,D,Q,O,T,R
19000	18094	9	89823	73447	F,N,D,Q,O,T,R,S,I
20000	19943	10	98426	76943	F,N,D,Q,O,T,R,S,I,K
21000	19943	10	98426	76943	F,N,D,Q,O,T,R,S,I,K
22000	19943	10	98426	76943	F,N,D,Q,O,T,R,S,I,K
23000	22424	11	106910	80246	F,N,D,Q,O,T,R,S,I,K,G
24000	23841	12	114547	83095	F,N,D,Q,O,T,R,S,I,K,G,M

Segment Prioritization Objective Function

Optimization Results

Capacity	Number of Units Received	Number of Containers Selected	Number of Matched Units	O.F. Value	Containers Selected
10000	9875	5	7953	48475	I,J,O,P,S
11000	10925	6	8483	53146	G,I,H,J,P,S
12000	11742	6	9124	57152	G,I,O,Q,P,S
13000	12877	7	9772	60402	G,I,H,K,J,P,S
14000	13943	7	10716	66069	G,I,J,O,Q,P,S
15000	14855	8	10913	67779	B,G,I,H,J,Q,P,S
16000	15792	8	11819	72639	G,I,H,J,O,Q,P,S
17000	16866	9	12565	74833	E,G,I,H,J,O,Q,P,S
18000	17744	9	12914	77955	G,I,H,K,J,O,Q,P,S
19000	18818	10	13660	80149	E,G,I,H,K,J,O,Q,P,S
20000	19797	10	14133	82410	D,G,I,H,J,M,O,Q,P,S
21000	20725	10	14635	85304	D,G,F,I,H,J,O,Q,P,S
22000	21799	11	15357	87402	E,D,G,F,I,H,J,O,Q,P,S
23000	22739	11	15532	89320	D,G,F,I,H,J,O,Q,P,S,R
24000	23813	12	16203	91163	E,D,G,F,I,H,J,O,Q,P,S,R

Heuristic Results

Capacity	Number of Units Received	Number of Containers Selected	Heuristic Score	Calculated O.F. Value	Containers Selected
10000	9930	4	5072	44907	P,Q,O,F,
11000	9930	4	5072	44907	P,Q,O,F,
12000	9930	4	5072	44907	P,Q,O,F,
13000	12281	5	23072	53640	P,Q,O,F,D
14000	12281	5	23072	53640	P,Q,O,F,D

15000	14233	6	27029	59532	P,Q,O,F,D,K
16000	14233	6	27029	59532	P,Q,O,F,D,K
17000	16560	7	30482	65434	P,Q,O,F,D,K,N
18000	16560	7	30482	65434	P,Q,O,F,D,K,N
19000	18761	8	33922	73214	P,Q,O,F,D,K,N,J
20000	18761	8	33922	73214	P,Q,O,F,D,K,N,J
21000	20775	9	37304	77111	P,Q,O,F,D,K,N,J,R
22000	20775	9	37304	77111	P,Q,O,F,D,K,N,J,R
23000	22624	10	40399	81821	P,Q,O,F,D,K,N,J,R,H
24000	22624	10	40399	81821	P,Q,O,F,D,K,N,J,R,H

Balanced Objective Function

Optimization Results

Capac-ity	Number of Units Received	Number of Con-tainers Selected	Number of Matche d Units	Dollars of Matche d Units	O.F. Value	Containers Selected
10000	9910	6	7647	44437	262305	E,G,I,H,Q,S
11000	10921	6	8214	50634	280869	D,G,I,H,N,S
12000	11995	7	9066	55668	302475	E,D,G,I,H,N,S
13000	12915	7	9732	58351	314574	G,I,H,N,Q,S,T
14000	13574	8	9821	57686	332638	B,E,G,I,H,N,Q,S
15000	14588	8	10642	64015	351969	E,D,G,I,H,N,Q,S
16000	15588	9	11161	65528	368541	B,E,G,I,H,N,Q,S,R
17000	16602	9	11826	71335	383942	E,D,G,I,H,N,Q,S,R
18000	17789	10	12653	73225	399258	B,E,G,I,H,J,N,Q,S,R
19000	18803	10	13318	79032	414659	E,D,G,I,H,J,N,Q,S,R
20000	19741	11	13570	76865	417893	B,E,G,I,H,K,J,N,Q,S, R
21000	20209	11	14143	82047	426601	E,D,G,I,H,J,L,N,Q,S, R
22000	21962	11	15168	85971	439223	D,G,I,H,J,N,Q,P,S,R, T
23000	22690	12	15442	86960	447602	E,D,G,I,H,J,L,N,Q,P, S,R
24000	23866	12	15990	90685	458441	E,D,G,F,I,H,J,N,Q,P, S,R

Heuristic Results

Capacity	Number of Units Received	Number of Containers Selected	Heuristic Score	Calculated O.F. Value	Containers Selected
10000	9853	4	95894	214755	F,Q,D,N
11000	9853	4	95894	214755	F,Q,D,N
12000	9853	4	95894	214755	F,Q,D,N
13000	12127	5	117021	239623	F,Q,D,N,O
14000	13976	6	135516	281701	F,Q,D,N,O,H
15000	13976	6	135516	281701	F,Q,D,N,O,H
16000	15990	7	153897	315229	F,Q,D,N,O,H,R
17000	15990	7	153897	315229	F,Q,D,N,O,H,R
18000	15990	7	153897	315229	F,Q,D,N,O,H,R
19000	18191	8	171273	345235	F,Q,D,N,O,H,R,J
20000	19608	9	188579	385787	F,Q,D,N,O,H,R,J,S
21000	19608	9	188579	385787	F,Q,D,N,O,H,R,J,S
22000	21360	10	204953	396761	F,Q,D,N,O,H,R,J,S,T
23000	22862	11	221291	426946	F,Q,D,N,O,H,R,J,S,T,I
24000	22862	11	221291	426946	F,Q,D,N,O,H,R,J,S,T,I

10 Results Discussion

The optimization results and heuristic results can be compared to determine the potential improvement a mathematical optimization model has over a heuristic. The analysis of the results and discussion is broken into the following points:

Receiving Capacity Limits

Making an assumption that the optimization engine is selecting containers as best as possible and the heuristic is only a best effort by humans to mimic the optimization, the difference between the units received is one sign of the heuristic's performance.

Each objective function works to select containers which maximize the objective function up to the receiving capacity limit. By selecting different sets of containers for each receiving capacity limit, the engine ensures that it can incrementally improve the objective function as much as possible, while taking into account the fact that credit towards the maximization should not be given to collateral product which is received. Therefore, an analysis of the receiving capacity limits is the best proxy for comparison at this point. When the heuristic rules select containers whose sum of product inside is less than what the optimization engine selects, it is a sign that there were better containers to select to maximize the objective function. Note the heuristic works in a step function manner, as opposed to the optimization engine which is more linear due to it selecting different sets of containers for receipt at each receiving capacity constraint.

If the heuristic rules select containers whose sum of product inside is greater than what the optimization engine selects, it means there is additional product being brought into the facility. This additional product is in the form of a) more units than what the refined demand signal r_p states should be received and is therefore taking up inventory space, b) collateral product which is not needed, or c) a combination of these two. In other words, the optimization engine elected to select fewer units to maximize the objective function and is therefore more efficient with its selection decisions.

Throughput Objective Function:

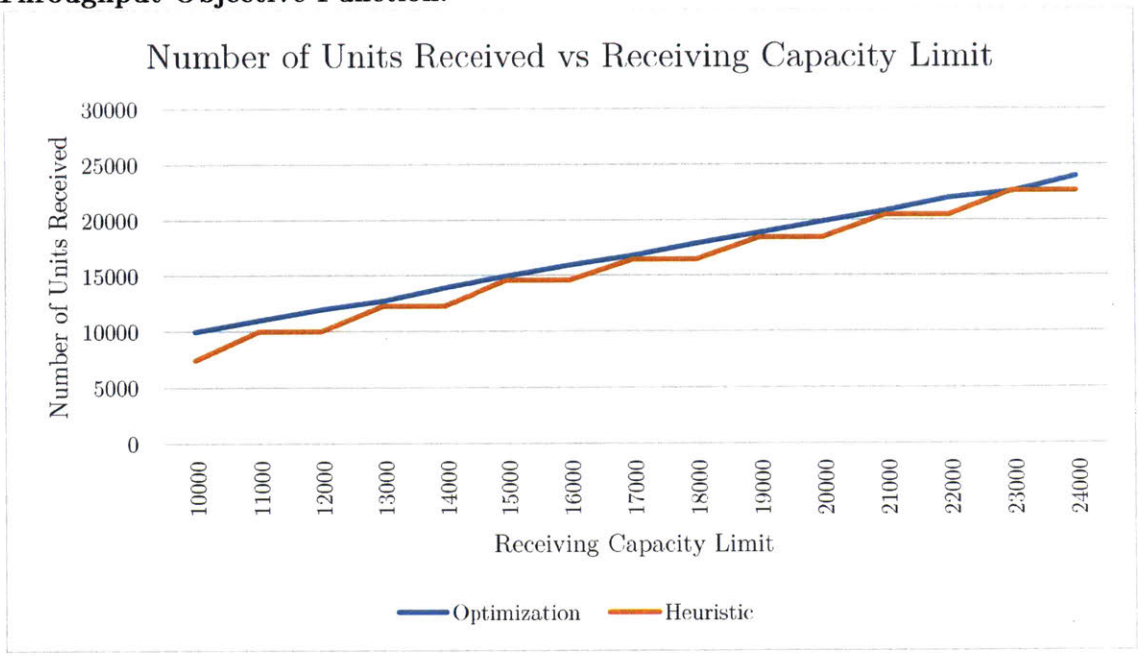

Figure 3 Throughput Obj. Function Units Received vs Receiving Capacity

The optimization engine leaves an average difference of 125 units between the receiving capacity limit and the units received, whereas the heuristic leaves an average of 1200 units difference.

Revenue Objective Function:

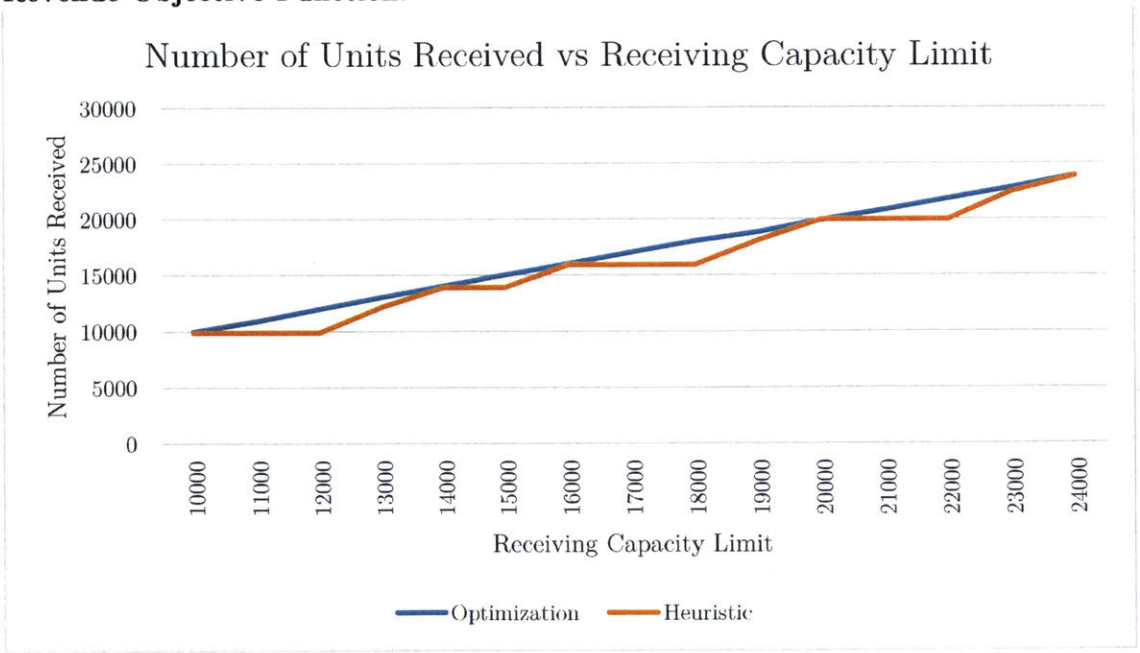

Figure 4 Revenue Obj. Function Units Received vs Receiving Capacity

The optimization engine leaves an average difference of 96 units between the receiving capacity limit and the units received, whereas the heuristic leaves an average of 913 units difference.

Segment Prioritization Objective Function:

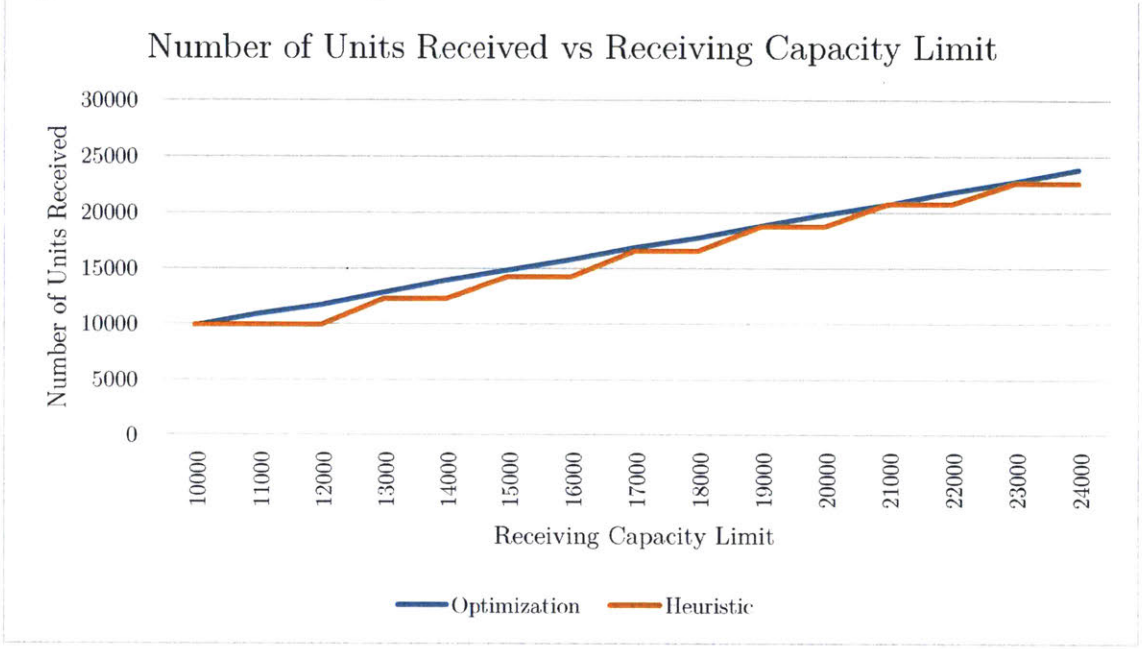

Figure 5 Segment Obj. Function Units Received vs Receiving Capacity

The optimization engine leaves an average difference of 179 units between the receiving capacity limit and the units received, whereas the heuristic leaves an average of 983 units difference.

Balanced Objective Function:

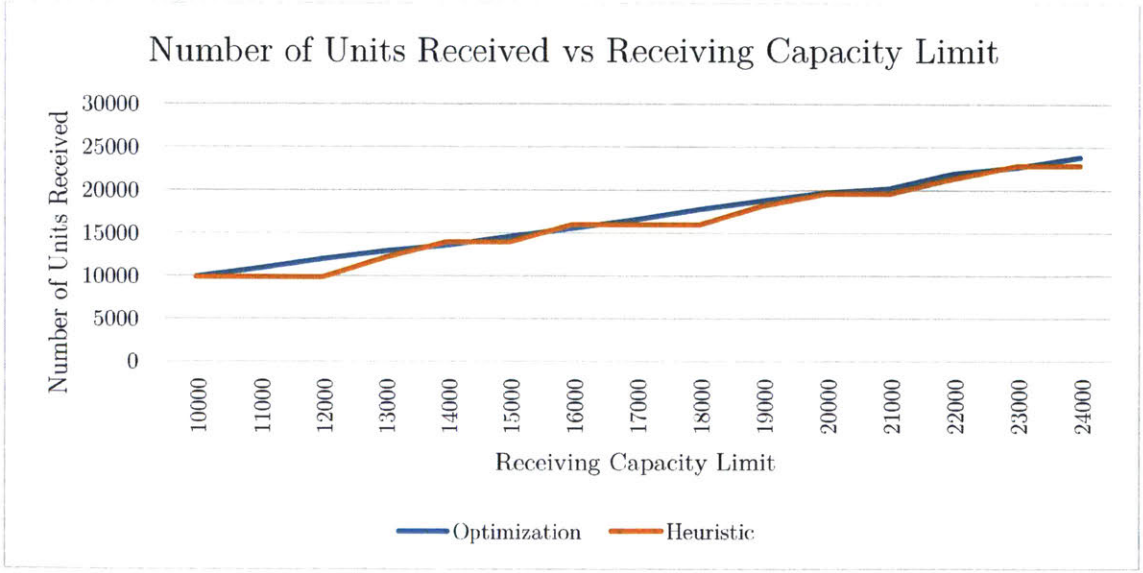

Figure 6 Balanced Obj. Function Units Received vs Receiving Capacity

The optimization engine leaves an average difference of 256 units between the receiving capacity limit and the units received, whereas the heuristic leaves an average of 860 units difference.

Optimization and Heuristic Receiving Limits

In this example problem, there is a greater sum of available SKU's (36,946) compared to the sum of SKU's demanded (23,264). Because the "Collateral Product Prevention Constraint" and the "Quantity of Matched SKU's Cannot Exceed Quantity of Received SKU's Constraint" must be relaxed when using the heuristic rules and each container has a non-negative score associated with it, the heuristic rules will select every container available if there is a receiving capacity limit equal to or greater than 36,946 units. Given the same receiving capacity constraint, the optimization engine will stop selecting containers when $y_p \geq r_p$. This means that once the optimization engine picks sufficient containers to meet the refined demand signal, it will stop selecting containers because selecting additional containers won't add to the objective function.

However, if the the receiving capacity is greater than the number of units available for receipt, i.e. there is no container backlog, the heuristic may be an acceptable means to select containers. The intuition behind this is that if there is no backlog of containers, all containers have some demand against them meaning a positive container score, and all the containers can be received, the optimization engine and heuristic will yield the same results: receive every container.

Objective Function Comparison and Curves

The plots below show the objective function values for the optimization engines and the calculated objective function values for the heuristics. For each objective function, the optimization engine outperformed the heuristic at each receiving capacity value.

In each optimization engine run, the value of the objective function acts in a nonlinear way. The objective function tapers off in a logarithmic fashion as the receiving capacity increases. This is because when $y_p \geq r_p$, y_p no longer contributes to the objective function. This fact ties into the description above regarding how the optimization engine will limit itself to containers received once it meets the required demand.

The fact that this happens makes another case for use of the lean optimization model combined with the optimization engine - given a large set of containers available for receipt and an inventory shelf space capacity constraint in the DC, there are diminishing returns to the objective function when receiving containers. Once the objective function tapers off, there is no benefit to receiving additional containers as this product will sit in inventory.

Below are the graphs of the objective functions against receiving capacity. While the graphs may look somewhat linear, the graphs can be compared against the tables in the results section above to confirm the non-linearity.

Throughput Objective Function:

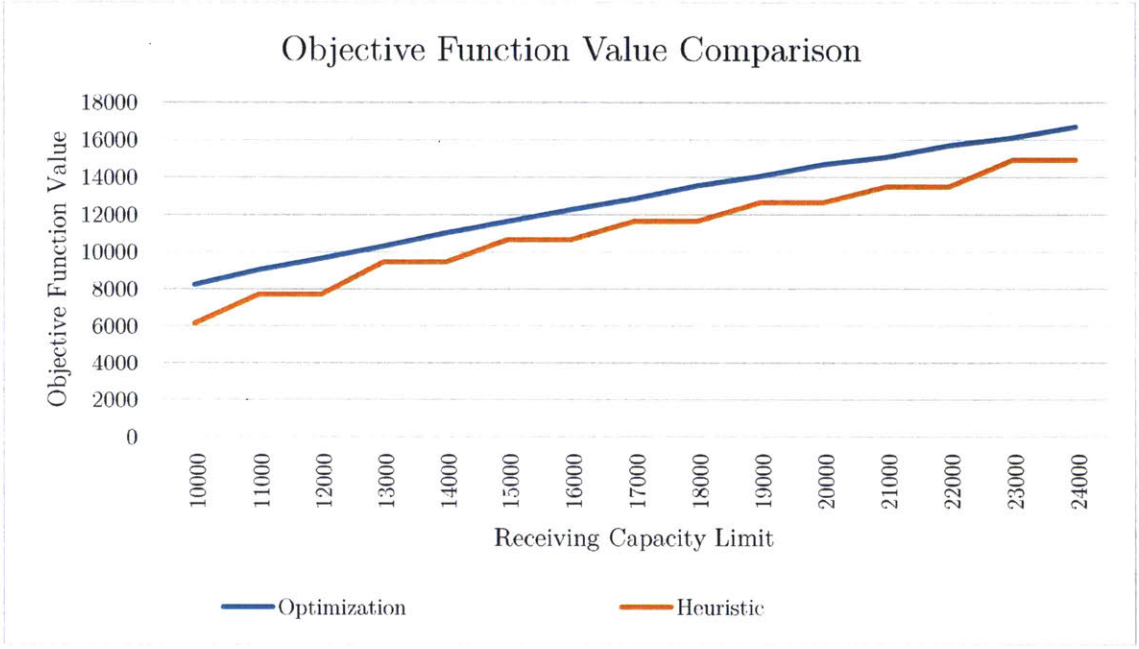

Figure 7 Throughput Obj. Function Value vs Receiving Capacity

Revenue Objective Function:

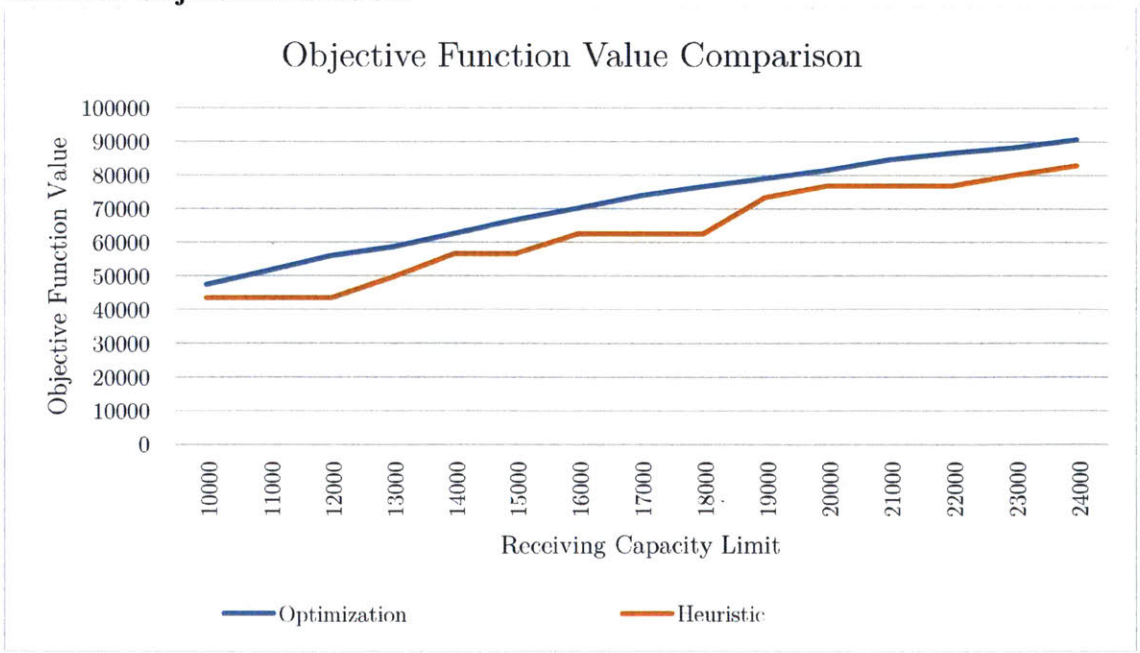

Figure 8 Revenue Obj. Function Value vs Receiving Capacity

Segment Prioritization Objective Function:

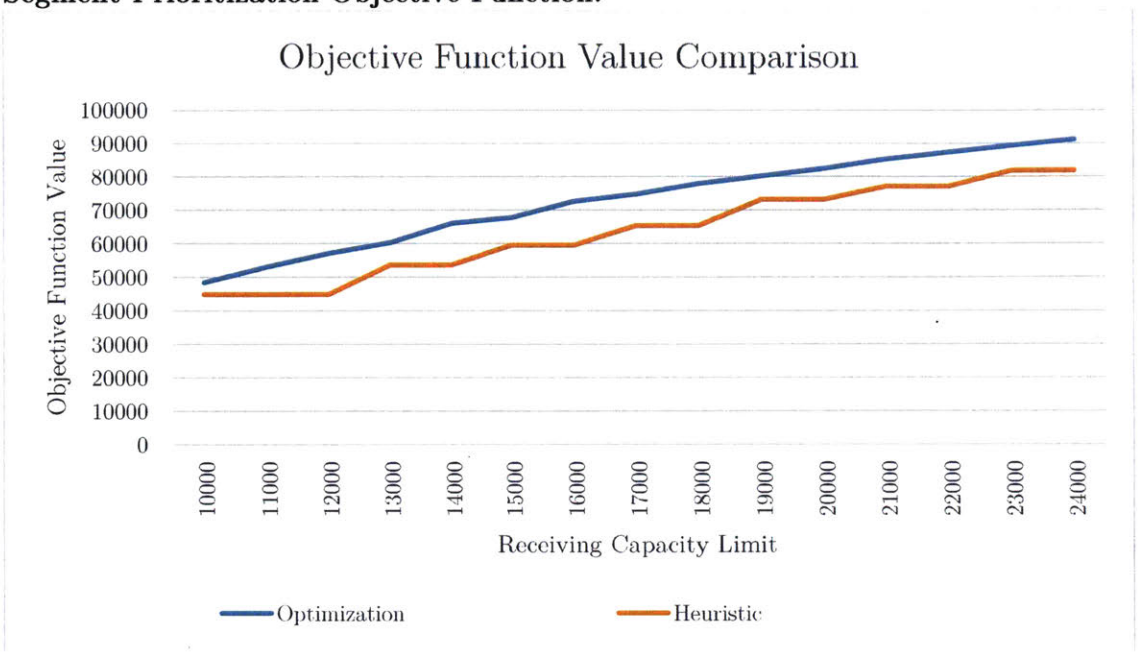

Figure 9 Segment Prioritization Obj. Function Value vs Receiving Capacity

Balanced Objective Function:

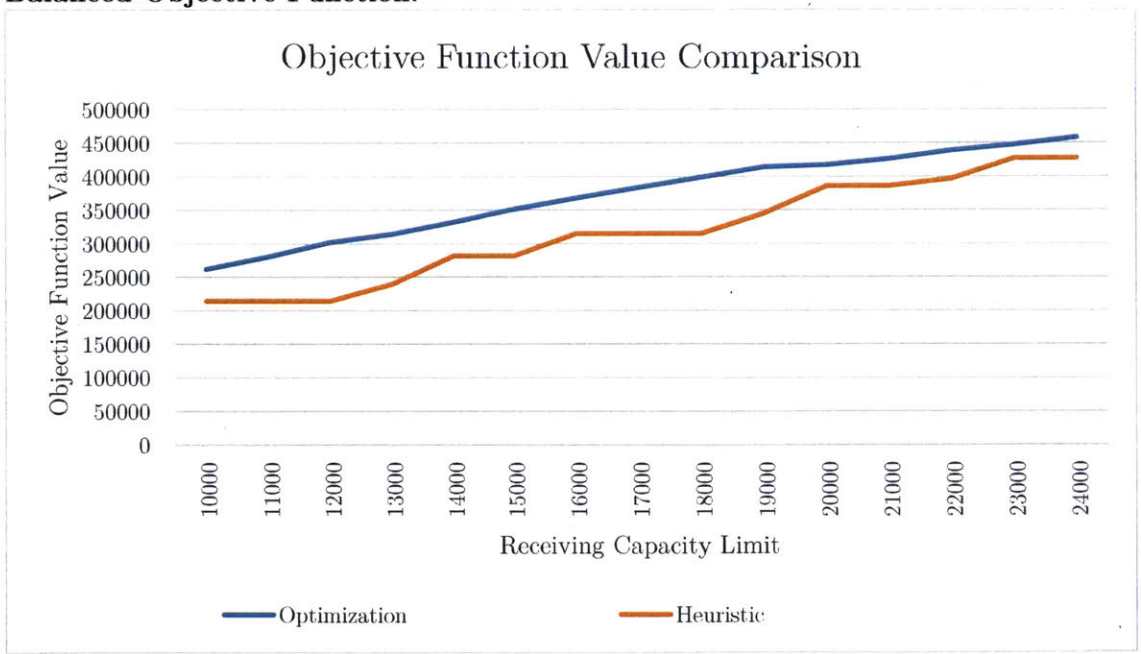

Figure 10 Segment Prioritization Obj. Function Value vs Receiving Capacity

11 Summary and Conclusion

This thesis analyzes the challenges associated with long lead times in supply chains when operating from the perspective of a DC needing to make decisions of which containers to receive on a daily basis. The size of the supply chain and length of time product spends traveling through that supply chain invites opportunities for changes in the supply and demand characteristics of orders. This manifests itself once the containers reach the DC and the quantities of product inside the container do not match the quantities of product required for receipt.

The thesis details how this timing mismatch and the choice of containers selected for receipt can create storage capacity constraints in the DC. These storage capacity constraints can lead to throughput capacity constraints reducing the ability to deliver product in full and on time. As the speed at which product flows through a DC is reduced, there is a drop in customer service and revenue generated. An investigation was performed into the current method of selecting containers for receipt, and a recommendation for a clean slate approach to this process was made.

The clean slate approach is a lean inventory management system which is used to calculate all outbound demand from the DC for product within a certain time window. The resultant demand is called the refined demand signal and is a key input into a mathematical container selection optimization engine. The optimization engine allows the user to select an objective function and a receiving capacity constraint, then selects a set of containers to receive. This optimization engine maximizes the receipt effectiveness of the DC.

A heuristic is created to provide a comparison against the optimization engine, and the results of both are displayed and discussed. There are several parameters where the optimization engine outperforms the heuristic. As a summary, the optimization engine is more effective at selecting containers for receipt than the heuristic when the receiving capacity constraint is less than the sum of products across all containers. When the sum of the product in the containers available for receipt is less than the receiving capacity constraint, the containers chosen by the optimization engine and by the heuristic will be the same, *assuming there is sufficient demand for product in all the containers.* If the demand for product inside the containers is less than the available supply, the optimization engine will stop selecting containers once the refined demand signal has been satisfied. This prevents additional product from entering the DC and adding to the space constraint problem.

The conclusion is that, given the optimization engine runs in under a minute and is more accurate at maximizing the receipt effectiveness over the heuristic, the optimization engine is superior to the heuristic.

11.1 Further Research Opportunities

The most obvious improvement on this model would be to refine it so the selection of containers is adjusted to maximize outbound order shipments. Orders will not ship until every line item is filled. The current iteration of the model focuses only on maximizing the filling of line items relying on the law of large numbers to complete an order. An opportunity exists to incorporate logic which will select containers based on filling an order. Say there's a very large order which can be completed by receiving 5 units of a small red shirt. A container that has that 5 or more units of that small red shirt should prioritized for receipt to complete that order so it can ship. In the current model there is no way to ensure that container is prioritized - it treats the refined demand signal for all orders the same.

There are additional opportunities for optimization programming upstream of the DC. One example is the repacking of containers at the various nodes in the supply chain. The current method of repacking containers is similar to the current method of receiving containers - multiple systems are being used to create a heuristic for determining how containers should be sorted and packed. Repacking containers in a way that maximizes the demand by date for product in the container would assist with the selection of containers at the DC because there would ostensibly be greater demand against those containers and they would have fewer collateral products inside.

Data can also be gathered about potential stock-outs through the use of the optimization engine. The potential stock-outs here are defined as product that is needed within the time window defined but not located in upstream containers in the supply chain which will be at the DC within the time window. Identifying these potential stock-outs would assist with deciding which shipments to expedite. Both of these opportunities would assist with reducing costs within the supply chain, improve on-time delivery performance, and increase revenue.

This page intentionally left blank

12 Appendix

12.1 Appendix 1

Receipt Requirements:

The table below shows the results of the lean inventory model which are an important input to the optimization engine. This is the refined demand signal r_p . The table shows the SKU's which are required, the quantity of the SKU required, the priority segment of the SKU, and the Revenue Scalar for the SKU as well.

SKU: A SKU is in the form style-color-size. Three random letters represent the style, three random numbers represent the color, and 3 sizes (S,M,L) were randomly assigned to the style-color level. This is representative of the letter p.

Priority Segment: A scalar of 1, 5, or 10 was randomly assigned to each SKU. This is representative of the α_p value.

Revenue Scalar: A 1-10 scalar was randomly assigned to each SKU. This is representative of the β_p value.

Required Quantity: The quantity of a SKU required, this represents the refined demand signal for a SKU, or r_p

SKU	Priority Segment	Required Quantity	Revenue Scalar	SKU	Priority Segment	Required Quantity	Revenue Scalar
LDM-027-M	5	151	4	OGO-605-S	10	232	2
VKA-848-L	5	152	7	FRC-734-M	10	231	1
IDF-637-L	10	431	2	OIY-282-M	5	449	2
RIN-008-M	5	193	9	LBE-411-M	1	510	6
SRP-593-M	10	339	2	PXU-047-S	1	427	9
SAK-032-S	1	420	1	CKL-415-M	5	217	3
UEZ-786-S	1	240	9	GSD-070-L	1	90	10

OVM-167-L	1	475	1	ZZN-336-S	10	81	4
IXK-292-S	10	78	5	VDF-755-L	5	274	10
JXT-218-S	5	111	5	VWH-814-L	1	411	3
UWS-919-S	10	357	7	ULH-792-M	10	212	3
AUK-379-M	10	82	10	LIG-680-L	1	205	6
VDS-969-L	10	79	3	HQU-318-S	5	306	10
AHZ-105-S	10	480	9	KKI-226-L	5	459	8
ZMX-658-M	5	593	5	AQV-593-M	10	549	1
DYL-193-M	5	263	8	FZY-938-M	10	205	9
EWA-766-M	1	188	2	SFM-900-L	5	207	9
NNI-623-S	5	588	1	MCZ-559-M	5	59	2
BAM-368-L	5	285	2	CNJ-577-S	5	224	6
MOU-793-M	5	284	2	WSS-621-S	5	67	9
YAG-862-L	1	281	1	QRS-147-M	5	236	2
FZT-702-M	1	322	10	VSU-491-M	5	600	7
RAV-208-M	10	538	8	EPL-030-L	10	359	2
PSP-404-L	1	581	8	OGF-357-S	1	240	5
KBU-096-S	5	477	5	EHV-036-L	1	247	7
YQD-135-L	10	206	3	JFL-644-S	5	330	6
VBF-161-L	10	227	5	UZR-765-S	10	494	1
LIV-417-M	1	530	3	UFO-537-S	1	285	10
CJL-756-M	5	151	2	WVH-814-M	1	513	10

NCS-004-M	5	241	1	PIS-647-L	1	168	2
FKO-844-L	10	159	1	SDD-242-L	5	446	7
XNW-753-S	1	304	7	UQF-821-S	5	231	8
ZBU-075-M	10	487	4	OSX-733-S	5	400	4
GYV-716-S	5	71	7	VMC-087-L	1	525	4
OAL-071-S	5	358	8	ICH-875-S	5	466	6
XUP-573-L	5	242	6	TLX-958-L	10	130	3
BPH-792-S	10	183	3	WPO-173-L	1	535	9
XER-756-M	10	497	9				

Container Supply:

The tables below show the contents of each container which is available for receipt. The list is ordered by container from A-T. The SKU's shown are the set of matched SKU's, set Y. All other SKU's which are in the container are collateral product and represented by the SKU "ABC-111-L". The quantity of each SKU is listed as well.

SKU: A SKU is in the form style-color-size. Three random letters represent the style, three random numbers represent the color, and 3 sizes (S,M,L) were randomly assigned to the style-color level. This is representated in the problem formulation by the letter "p".

Container: Each container is represented by a distinct letter A-T, meaning there are 20 containers available to the DC. Each container is represented in the optimization formulation by the letter "c".

Supply Quantity: The quantity of a SKU p supplied in container c, this represents the refined demand signal for a SKU, or q_{cp}

Container	SKU	q	Container	SKU	q	Container	SKU	q
A	VKA-848-L	182	I	RIN-008-M	29	O	MOU-793-M	111
A	GYV-716-S	196	I	IXK-292-S	70	O	CJL-756-M	154
A	ICH-875-S	172	I	VBF-161-L	162	O	FZY-938-M	89
A	MCZ-559-M	162	I	PIS-647-L	95	O	OGF-357-S	177
A	BAM-368-L	70	I	AQV-593-M	100	O	ZMX-658-M	125
A	LDM-027-M	17	I	SRP-593-M	31	O	EWA-766-M	66
A	FRC-734-M	160	I	OSX-733-S	95	O	GYV-716-S	117
A	SRP-593-M	20	I	MCZ-559-M	25	O	ZZN-336-S	45
A	LIV-417-M	19	I	UWS-919-S	62	O	UFO-537-S	87
A	ABC-111-L	349	I	XUP-573-L	169	O	UEZ-786-S	13
B	NCS-004-M	115	I	KKI-226-L	37	O	WPO-173-L	181
B	BPH-792-S	35	I	KBU-096-S	49	O	AQV-593-M	142
B	XER-756-M	169	I	OAL-071-S	140	O	IDF-637-L	65
B	VDS-969-L	116	I	HQU-318-S	136	O	AUK-379-M	182
B	VKA-848-L	190	I	CNJ-577-S	119	O	BAM-368-L	51
B	OGF-357-S	68	I	ABC-111-L	183	O	VBF-161-L	65
B	ZMX-658-M	14	J	UEZ-786-S	52	O	NCS-004-M	149
B	YAG-862-L	193	J	AHZ-105-S	134	O	BPH-792-S	100
B	TLX-958-L	72	J	KKI-226-L	36	O	EPL-030-L	143
B	AQV-593-M	46	J	EPL-030-L	171	O	JFL-644-S	91

B	ABC-111-L	319	J	IDF-637-L	25	O	ABC-111-L	121
C	SRP-593-M	191	J	PXU-047-S	39	P	PSP-404-L	189
C	OIY-282-M	35	J	VWH-814-L	39	P	ZBU-075-M	55
C	VWH-814-L	169	J	LIG-680-L	180	P	ULH-792-M	166
C	WSS-621-S	158	J	AQV-593-M	168	P	MOU-793-M	128
C	SAK-032-S	152	J	FZY-938-M	148	P	VMC-087-L	141
C	EWA-766-M	61	J	NNI-623-S	123	P	SRP-593-M	184
C	YQD-135-L	13	J	BAM-368-L	83	P	JXT-218-S	167
C	IXK-292-S	132	J	ZBU-075-M	67	P	VDS-969-L	28
C	ZBU-075-M	90	J	SDD-242-L	57	P	VBF-161-L	43
C	ABC-111-L	357	J	LBE-411-M	155	P	CKL-415-M	109
D	QRS-147-M	185	J	GSD-070-L	39	P	GSD-070-L	79
D	UFO-537-S	114	J	WVH-814-M	199	P	ZZN-336-S	156
D	FZT-702-M	186	J	ABC-111-L	486	P	JFL-644-S	166
D	YQD-135-L	114	K	NNI-623-S	165	P	UZR-765-S	170
D	XUP-573-L	41	K	FKO-844-L	141	P	CJL-756-M	71
D	VDF-755-L	131	K	GSD-070-L	40	P	FKO-844-L	102
D	IDF-637-L	178	K	ZZN-336-S	155	P	FRC-734-M	187
D	RIN-008-M	188	K	VDF-755-L	112	P	VDF-755-L	27
D	IXK-292-S	42	K	OGO-605-S	65	P	LIG-680-L	54
D	XNW-753-S	35	K	ULH-792-M	126	P	ABC-111-L	259
D	BPH-792-S	111	K	SDD-242-L	38	Q	IDF-637-L	90
D	HQU-318-S	111	K	ICH-875-S	177	Q	YQD-135-L	166
D	CNJ-577-S	102	K	OIY-282-M	189	Q	SDD-242-L	47
D	XER-756-M	139	K	QRS-147-M	114	Q	OSX-733-S	159
D	PIS-647-L	181	K	DYL-193-M	57	Q	OGO-605-S	158
D	VWH-814-L	178	K	EWA-766-M	60	Q	HQU-318-S	131
D	UZR-765-S	18	K	YQD-135-L	123	Q	UEZ-786-S	72
D	ABC-111-L	297	K	XUP-573-L	111	Q	IXK-292-S	19
E	SAK-032-S	145	K	EHV-036-L	101	Q	LIV-417-M	26
E	AUK-379-M	107	K	ABC-111-L	178	Q	PIS-647-L	196
E	EWA-766-M	53	L	ZMX-658-M	74	Q	LBE-411-M	96
E	KBU-096-S	12	L	LIV-417-M	174	Q	UZR-765-S	68
E	VBF-161-L	48	L	FRC-734-M	48	Q	ZMX-658-M	170
E	CKL-415-M	63	L	VSU-491-M	162	Q	RAV-208-M	66
E	QRS-147-M	69	L	JXT-218-S	71	Q	KBU-096-S	54
E	WVH-814-M	99	L	FKO-844-L	81	Q	XER-756-M	174
E	FZT-702-M	142	L	SFM-900-L	13	Q	ULH-792-M	79
E	UWS-919-S	44	L	OVM-167-L	40	Q	EHV-036-L	141
E	ICH-875-S	119	L	YAG-862-L	23	Q	RIN-008-M	77
E	ABC-111-L	173	L	LIG-680-L	19	Q	ZZN-336-S	175
F	JXT-218-S	10	L	LDM-027-M	21	Q	ABC-111-L	429
F	VDS-969-L	94	L	SAK-032-S	99	R	LDM-027-M	136

F	KBU-096-S	159	L	MOU-793-M	30	R	OAL-071-S	16
F	SFM-900-L	176	L	VMC-087-L	52	R	LIG-680-L	50
F	NNI-623-S	180	L	FZY-938-M	14	R	CNJ-577-S	134
F	ICH-875-S	192	L	WSS-621-S	42	R	AHZ-105-S	86
F	TLX-958-L	153	L	ABC-111-L	443	R	EHV-036-L	169
F	OIY-282-M	195	M	YAG-862-L	12	R	MOU-793-M	31
F	HQU-318-S	106	M	TLX-958-L	142	R	FKO-844-L	120
F	CNJ-577-S	52	M	PXU-047-S	171	R	GYV-716-S	191
F	WSS-621-S	172	M	CKL-415-M	57	R	UFO-537-S	107
F	AUK-379-M	169	M	SAK-032-S	144	R	XNW-753-S	43
F	DYL-193-M	132	M	JFL-644-S	11	R	ULH-792-M	125
F	PSP-404-L	138	M	PIS-647-L	176	R	KKI-226-L	179
F	OAL-071-S	52	M	OSX-733-S	198	R	QRS-147-M	120
F	VWH-814-L	33	M	JXT-218-S	92	R	VSU-491-M	149
F	WVH-814-M	44	M	NNI-623-S	187	R	OGF-357-S	108
F	GYV-716-S	25	M	FZT-702-M	176	R	ABC-111-L	250
F	WPO-173-L	152	M	OGO-605-S	91	S	LDM-027-M	119
F	MCZ-559-M	31	M	ABC-111-L	197	S	RIN-008-M	42
F	ABC-111-L	317	N	BAM-368-L	44	S	UWS-919-S	113
G	DYL-193-M	129	N	UQF-821-S	144	S	DYL-193-M	150
G	XNW-753-S	16	N	WPO-173-L	68	S	BPH-792-S	133
G	JFL-644-S	98	N	UZR-765-S	76	S	XER-756-M	155
G	UQF-821-S	126	N	WVH-814-M	17	S	VSU-491-M	155
G	WPO-173-L	14	N	OVM-167-L	12	S	EPL-030-L	44
G	FRC-734-M	102	N	YAG-862-L	19	S	UEZ-786-S	137
G	TLX-958-L	84	N	RAV-208-M	168	S	RAV-208-M	74
G	SFM-900-L	17	N	PSP-404-L	182	S	ABC-111-L	295
G	VKA-848-L	132	N	GSD-070-L	126	T	OVM-167-L	97
G	OVM-167-L	25	N	KKI-226-L	73	T	FZT-702-M	165
G	VDS-969-L	153	N	OGF-357-S	174	T	LBE-411-M	93
G	AHZ-105-S	144	N	CJL-756-M	144	T	EHV-036-L	53
G	OIY-282-M	136	N	NCS-004-M	47	T	VKA-848-L	18
G	ABC-111-L	299	N	LBE-411-M	106	T	AUK-379-M	152
H	UWS-919-S	63	N	FZY-938-M	195	T	ZBU-075-M	24
H	RAV-208-M	176	N	MCZ-559-M	98	T	SDD-242-L	22
H	XUP-573-L	31	N	SFM-900-L	166	T	EPL-030-L	194
H	VMC-087-L	66	N	ABC-111-L	468	T	LIV-417-M	187
H	CJL-756-M	169				T	UQF-821-S	151
H	NCS-004-M	146				T	VMC-087-L	11
H	XNW-753-S	46				T	PXU-047-S	89
H	OAL-071-S	115				T	PSP-404-L	31
H	VSU-491-M	93				T	OSX-733-S	83
H	AHZ-105-S	42				T	CKL-415-M	87

H	OGO-605-S	130
H	VDF-755-L	132
H	WSS-621-S	117
H	UQF-821-S	38
H	PXU-047-S	68
H	ABC-111-L	417
T	UFO-537-S	187
T	ABC-111-L	108

A summary list of the total quantity of products in each container is below:

Container	$\sum_{p=1}^{P} q_{pc}$
A	1347
B	1337
C	1358
D	2351
E	1074
F	2582
G	1475
H	1849
I	1502
J	2201
K	1952
L	1406
M	1654
N	2327
O	2274
P	2481
Q	2593
R	2014
S	1417
T	1752

12.2 Appendix 2

Heuristic Calculations:

This is a list of the calculated heuristic score for each container for determining the order of containers which are selected for receipt.

Container	Units	Heuristic Throughput Score	Heuristic Revenue Score	Heuristic Segment Score	Heuristic Balanced Score
A	1347	1347	4467	2157	9077
B	1337	1337	4284	2213	10271
C	1358	1358	3714	2046	8088
D	2351	2351	11991	4016	22351
E	1074	1074	5380	1562	9717
F	2582	2582	14222	4579	27864
G	1475	1475	6349	2780	14544
H	1849	1849	8603	3095	18495
I	1502	1502	7386	2968	16338
J	2201	1715	9668	3440	17376
K	1952	1774	7206	3957	14823
L	1406	963	3759	1662	6767
M	1654	1457	6091	2468	8983
N	2327	1859	12059	3453	22258
O	2274	2153	10388	4613	21127
P	2481	2222	8484	5072	16246
Q	2593	2164	10957	4792	23421
R	2014	1764	10253	3382	18381
S	1417	1122	7637	2626	17306
T	1752	1644	10285	2745	16374

13 References

Knight, Philip H. *Shoe Dog: A Memoir By the Creator of Nike.* First Scribner hardcover edition. Scribner, 2016.

Willems, Sean. 2013. "Inventory Optimization: Evolving from Fad to Necessity." *Supply Chain Management Review* March/April 2013: 10-17. PDF Jul. 2015

Burke, Edmund and Kendall, Graham. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.* Second Edition. Spring Science and Business Media, New York, 2014.

Ferguson, Thomas. *Linear Programming: A Concise Introduction.* PDF file: https://www.math.ucla.edu/~tom/LP.pdf , accessed Web. 18 Mar. 2017.

Cachon, Terwiesch. *Matching Supply With Demand: An Introduction to Operations Management.* Second Edition. McGraw-Hill/Irwin, 2009.

Gabris, Andrew. 2016. "Size Curve Optimization for Replenishment Products". LGO Thesis, Massachusetts Institute of Technology.

United States Securities and Exchange Commission: Nike, Inc. 10-K https://www.sec.gov/Archives/edgar/data/320187/000032018716000336/nke-5312016x10k.htm Accessed 8 April 2017.

Lurie, Amanda. 2016. "Accelerating the Onboarding of a New Factory Partner". LGO Thesis, Massachusetts Institute of Technology.

Bradley, Hax, Magnanti. *Applied Mathematical Programming.* First Edition. Addison-Wesley Publishing Company, Inc.

Simchi-Levi, David. Class Notes from 15.762: Supply Chain Planning. Massachusetts Institute of Technology. Spring 2016.

Willems, Sean. *Analytical Consult Methods to Solve Supply Chain Problems.* Copyright January 2016. PDF File Accessed 2 February 2017.