ABSTRACT

ONLINE FULFILLMENT: F-WAREHOUSE ORDER CONSOLIDATION AND BOPS STORE PICKING PROBLEMS

by
Wen Zhu

Fulfillment of online retail orders is a critical challenge for retailers since the legacy infrastructure and control methods are ill suited for online retail. The primary performance goal of online fulfillment is speed or fast fulfillment, requiring received orders to be shipped or ready for pickup within a few hours. Several novel numerical problems characterize fast fulfillment operations and this research solves two such problems. Order fulfillment warehouses (F-Warehouses) are a critical component of the physical internet behind online retail supply chains. Two key distinguishing features of an F-Warehouse are (i) Explosive Storage Policy – A unique item can be stored simultaneously in multiple bin locations dispersed through the warehouse, and (ii) Commingled Bins – A bin can stock several different items simultaneously. The inventory dispersion profile of an item is therefore temporal and non-repetitive. The order arrival process is continuous, and each order consists of one or more items. From the set of pending orders, efficient picking lists of 10-15 items are generated. A picklist of items is collected in a tote, which is then transported to a packaging station, where items belonging to the same order are consolidated into a shipment package. There are multiple such stations.

This research formulates and solves the order consolidation problem. At any time, a batch of totes are to be processed through several available order packaging stations. Tote assignment to a station will determine whether an order will be shipped in a single package or multiple packages. Reduced shipping costs are a key operational goal of an online retailer, and the number of packages is a determining factor. The decision variable is which station a tote should be assigned to, and the

performance objective is to minimize the number of packages and balance the packaging station workload. This research first formulates the order consolidation problem as a mixed integer programming model, and then develops two fast heuristics (#1 and #2) plus two clustering algorithm derived solutions. For small problems, the heuristic #2 is on average within 4.1% of the optimal solution. For larger problems heuristic #2 outperforms all other algorithms. Performance behavior of heuristic #2 is further studied as a function of several characteristics.

S-Strategy fulfillment is a store-based solution for fulfilling online customer orders. The S-Strategy is driven by two key motivations, first, retailers have a network of stores where the inventory is already dispersed, and second, the expectation is that forward positioned inventory could be faster and more economical than a warehouse based F-Strategy. Orders are picked from store inventory and then the customer picks up from the store (BOPS). A BOPS store has two distinguishing features (i) In addition to shelf stock, the layout includes a space constrained back stock of selected items, and (ii) a set of dedicated pickers who are scheduled to fulfill orders. This research solves two BOFS related problems: (i) Back stock strategy: Assignment of items located in the back stock and (ii) Picker scheduling: Effect of numbers of picker and work hours. A continuous flow of incoming orders is assumed for both problems and the objective is fulfillment time and labor cost minimization. For the back-stock problem an assignment rule based on order frequency, forward location and order basket correlations achieves a 17.6% improvement over a no back-stock store, while a rule based only on order frequency achieves a 12.4% improvement. Additional experiments across a range of order baskets are reported.

ONLINE FULFILLMENT: F-WAREHOUSE ORDER CONSOLIDATION AND BOPS STORE PICKING PROBLEMS

by
Wen Zhu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Industrial Engineering

Department of Mechanical and Industrial Engineering

December 2020

APPROVAL PAGE

ONLINE FULFILLMENT: F-WAREHOUSE ORDER CONSOLIDATION AND BOPS STORE PICKING PROBLEMS

Wen Zhu

Dr. Sanchoy K. Das, Dissertation Advisor Date
Professor of Mechanical and Industrial Engineering, NJIT

Dr. Athanassios Bladikas, Committee Member Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Esra Buyuktahtakin-Toy, Committee Member Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Wenbo Cai, Committee Member Date
Associate Professor of Mechanical and Industrial Engineering, NJIT

Dr. Junmin Shi, Committee Member Date
Associate Professor of Supply Chain Management and Finance, Martin Tuchman School of Management, NJIT

BIOGRAPHICAL SKETCH

Author:	Wen Zhu
Degree:	Doctor of Philosophy
Date:	December 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Industrial Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2020

- Master of Science in Healthcare Systems Management,
  New Jersey Institute of Technology, Newark, NJ, 2014

- Bachelor of Medicine in Preventive Medicine,
  Southwest Medical University, Luzhou, Sichuan, P.R. China, 2010

Major: Industrial Engineering

Presentations and Publications:

Zhu, W., and Das, S., Buy Online Fulfill from Store - Design and Control of Order Picking Operations, Conference Presentation, Institute for Operations Research and the Management Science Annual Conference Annual Meeting, Seattle, WA, October 2019

Zhu, W., and Das, S., Dynamic Consolidation of Picked Orders in an Online Orders Fulfillment Warehouse, Conference Presentation, Institute for Operations Research and the Management Science Annual Conference Annual Meeting, Seattle, WA, October 2019

Zhu, W., Helminsky, A., and Das, S., Buy Online, Fulfill from Store – Location Assignment and Order Picking, Conference Presentation, Production and Operations Management Society Annual Conference, Washington, D.C., May 2019

Zhu, W., and Das, S., Consolidation of Picked Orders in a Fulfillment Center with Explosive Storage, Poster Presentation, Institute for Operations Research and the Management Science Annual Conference Annual Meeting, Phoenix, AZ, November 2018

This dissertation is dedicated to my beloved family:

my husband 张敏

my father 朱贵森

my mother 郭乃兰

for their endless and unconditional love.

ACKNOWLEDGMENT

I would like to express my sincere gratitude to my advisor Professor Sanchoy Das for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this dissertation. Without him I would not have been able to complete this research, and without him I would not have made it through my Ph.D. study.

Besides my advisor, I would like to thank the rest of my dissertation committee: Dr. Athanassios Bladikas, Dr. Wenbo Cai, Dr. Esra Buyuktahtakin-Toy, and Dr. Junmin Shi, for their insightful comments and encouragement.

My sincere thanks to the Department of Mechanical & Industrial Engineering for the financial support; and to my colleagues: Dr. Jingran Zhang and Dr. Sevilay Onal, for all the support in my Ph.D. years.

Finally, I would like to thank my family: my husband, Min Zhang, my parents, Guisen Zhu and Nailan Guo for all the endless and unconditional support.

TABLE OF CONTENTS

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF TABLES
(Continued)

Table **Page**

LIST OF TABLES
(Continued)

LIST OF FIGURES

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Research Background

Online and internet retail is dramatically changing the global retail industry. Companies vested in traditional supply chains are being challenged by a new and rapidly evolving breed of fulfillment centers that are focused exclusively on online or internet retail. Das (2020) describes eight paradigm shifts that have driven the growth of a fast fulfillment infrastructure. Three of these paradigms are of particular interest in this dissertation: (i) Online Shopping, (ii) Point-of-Use Delivery and (iii) The Warehouse is the Store. The consequence of these paradigms is that that customer no longer visits the store. Picking up items from store or warehouse inventory and then delivering it to the customer address is now the responsibility of the retailers. Progressively, we have seen the remodeling and extension of traditional supply chains to efficiently and economically perform these new functions. A central facility in online retail is a fulfillment center, in purely online retailer this is usually a warehouse, while for an Omni channel retailer this is frequently the store itself. The focus of this research is on the development of operations models which are applied to these fulfillment facilities.

Historically, fulfillment centers were developed for the mail order business, and fulfillment times were measured in weeks. Internet retailers compete with brick and mortar retailers on both the marketing side, where the goal is to sell a product virtually, and on the fulfillment side, where the goal is to provide delivery within a few days. The key infrastructure components of internet retail are a network of fulfillment warehouses (F-Warehouses) or centers and a parcel delivery network. Amazon with close to a 50% market share of all U.S. online sales is the most successful online retailer, and they have built over a 100 new fulfillment centers.

The success of Amazon confirms that a single channel strategy, pure online, can effectively meet the demand requirements of most customers. To expand into the online channel, most traditional retailers have built and are operating an online retail channel alongside their established physical stores. The largest U.S. retailers (Walmart and Target) are designing and building a store based or S-Strategy supply chain solution to fulfill online customer orders, this contrasts with the vast fulfillment center network or F-Strategy solution built by Amazon. In effect they have converted the store into a fulfillment center.

Then we have two key research questions. The first one is: are successful F-Warehouses, such as those built by Amazon, operating with design and control paradigms that are quite different from traditional warehouses? If so, what are these F-Warehouses differences and what are the accompanying product flow processes, storage strategies, material movements, and operations control models? Onal et al. (2017) were one of the first to report on F-Warehouses and demonstrate the fulfillment time performance advantages. Specifically, they identified and described how explosive storage policies are being used by F-Warehouses to achieve faster fulfillment. The second one is: Is an S-Strategy competitive? Some argue that S-Strategy solutions are unlikely to provide the needed efficiency gains.

This research includes both F-Strategy and S-Strategy. Chapters 3 and 4 investigate new warehousing models that are unique to online order fulfillment warehouses (F-Warehouses). These facilities process a very large number of customer orders, each of which are for a few units of product. We focus on the process flows, operational control models and decision optimization problems. Chapter 5 presents the operational structure of the S-Strategy, and two decisions models that are integral to this strategy.

1.2 Online Order Fulfillment Warehouses

Order fulfillment warehouses (F-Warehouses) are a critical component of the physical internet behind online retail supply chains. The primary performance goal of an F-Warehouse is fast fulfillment, requiring received orders to be shipped within a few hours. Onal et al. (2017) identified six key differentiators between an F-Warehouse and traditional warehouses or fulfillment centers. Together these differentiators provide the platform for the most successful online retailers to achieve their fast fulfillment objectives. These differentiators also show that a traditional warehouse cannot be easily transformed to an F-Warehouse, and significant structural and operational changes are required. The six differentiators are summarized here.

(i) Explosive Storage Policy – Incoming bulk inventory is exploded into a large number of small lots which ate then dispersed to storage locations throughout the warehouse, (ii) Very Large Number of Beehive Storage Locations – Storage is organized into small bins (1-3 cubic feet) as opposed to large bulk holding spaces, (iii) Bins with Commingled Items – One of the most radical differentiators, is that multiple items are simultaneously stored in an unorganized way in the same bin, (iv) Immediate Fulfillment Objective – Customer orders arrive continuously throughout the day and the goal is for same day shipment, (v) Short Picking Routes with Single Unit Picks – Most orders are only for only a single unit and the pick list retrieves several different items within a short pick zone, (vi) High Transaction Volumes with Total Digital Control – There is a much higher rate of store/pick movements per unit shipment, and all movements are modelled and instructed by a central controller.

1.3 Amazon Fulfillment Center

Amazon is the largest internet retailer in the world as measured by revenue and market capitalization, is also known for developing and managing F-Warehouses successfully. In 1997, Amazon launched its distribution network with two fulfillment centers in

Seattle and Delaware dealing directly with individual customer orders. This class of warehouse has been first called “fulfillment center”. Currently, with total over a 110 Million square feet space of facilities and 250,000 employees, Amazon operates over 250 distribution facilities around the world including fulfillment centers, returns centers, specialty centers, and redistribution centers. The Amazon U.S. and Canada fulfillment network consists of more than 100 fulfillment centers, and more than 125,000 full-time employees.

F-Warehouses have a high degree of conveying automation, and in this context there are two types: (i) Man-to-Part: Similar to a classical configuration in that storage racks are stationary and the worker moves to the bin location and (ii) Part- to-Man: The storage racks move, usually by a robot swarm, and bring the bin to the stationary worker. In a classical Part-to-Man, the pick occurs before the move, whereas in an F-Warehouse the entire rack is moved, and the pick is done after the move. Conveyance of product within the F-Warehouse occurs primarily in totes. Outside of the receiving area there are no pallet movements. Totes are designed for manual handling such that workers can easily lift a loaded tote. Totes have unique identification numbers so that SKU items in totes can be easily tracked.

The models Onal et al. (2017) presented are the result of an observational study of two Amazon fulfillment centers in the USA, one located in Indiana (1.2 Million sq. ft.) and the other in Delaware (0.9 Million sq. ft.). Both were of the Man-to-Part type and built in 2012, with approximately 2500 warehouse workers or associates. In the newer fulfillment centers, items are stored on pods and brought to pickers by robots (Kiva Systems). Author of this research visited a type Part-to-Man fulfillment center, which is 1 Million sq. ft., opened in 2015, located in Baltimore, Maryland.

Currently, Amazon operates a variety of different types of fulfillment and distribution centers in the United States including small sortable, large sortable, large non-sortable, specialty apparel and footwear, specialty small parts, returns processing centers, and 3PL outsourced facilities. This research focus on the small sortable fulfillment centers, which generally house smaller items that can all fit in one box/shipment (e.g., books, DVDs, watches, etc.). These products are best described as being less than 18” and can be placed into a conveyable tote. The product flows of fulfillment centers can be sequenced into three distinct process groups: (i) receiving and stocking (ii) order picking and consolidation and (iii) truck assignment and loading. While none of the processes are unique to F-Warehouses, they are operationally different due to the explosive storage policy. The Kiva robots are used in the stocking and picking process. This research focus on the consolidation operational process.

1.4 Online Order Fulfillment in a BOPS Retailer

S-Strategy fulfillment is a store-based solution for fulfilling online customer orders. The S-Strategy is driven by two key motivations, first, retailers have a network of stores where the inventory is already dispersed, and second, the expectation is that forward positioned inventory could be faster and more economical than a warehouse based F-Strategy. The brick-and-mortar retailers’ expansion into an online channel can be viewed as an extension or variation of omnichannel retail and is quite different from store-ending channels. Most importantly, an online customer makes product selections on a web catalog and in most cases will not visit a store. This requires the retail supply chain to additionally execute an order fulfillment process. An omnichannel retailer with physical stores receives online customer orders through its website. Orders are then directed to a specific company store, where the ordered items are picked from shelf inventory. Picked items are packaged and the package is either (i) Shipped to the customer address – BOFS or (ii) Picked up from the store by the customer – BOPS.

The first goal of the BOPS operation is immediate order fulfillment, allowing the retailer to provide a faster service rate than Amazon. BOPS is an expensive activity, requiring a picker to walk through the store inventory and fulfill a customer's inline order. The second goal is to minimize the order fulfillment cost.

1.5 Research Objectives and Accomplishments

Fulfillment of online retail orders is a critical challenge for retailers since the legacy infrastructure and control methods are ill suited for online retail. The primary performance goal of online fulfillment is speed or fast fulfillment, requiring received orders to be shipped or ready for pickup within a few hours. Several novel numerical problems characterize fast fulfillment operations and this research solves two such problems.

1.5.1 Formulate and Solve the Order Consolidation Problem as a MIP

Online order fulfillment warehouses, similar to those used by Amazon, typically operate with an explosive storage policy. That is, each item is stocked in multiple random locations dispersed throughout the warehouse. Orders are then picked and collected in totes which are assigned to one of many packaging stations. The order arrival process is continuous, and each order consists of one or more items. From the set of pending orders, efficient picking lists of 10-15 items are generated, which is commonly referred to as the picklist assignment problem. A picklist of items is collected in a tote, which is then transported to a packaging station, where items belonging to the same order are consolidated into a shipment package. There are multiple such stations.

There is a one-to-many relationship between customer orders and totes. This research formulates and solves the order consolidation problem. At any time, a batch of totes are to be processed through several available order packaging stations. Tote assignment to a station will determine whether an order will be shipped in a single package or multiple packages. Reduced shipping costs are a key operational goal of an online retailer, and the number of packages is a determining factor. The decision variable is which station a tote should be assigned to, and the performance objective is to minimize the number of packages and balance the packaging station workload.

1.5.1.1 Formulate a Totes Consolidation Objective. In a high-volume F-Warehouse, hundreds of picked totes are generated every hour and the conveying system can direct a tote to any one of the available consolidators. The consolidator is then instructed to pick SKU items from a specific tote to create a shipping box for a customer order. In the ideal case all SKUs for the order are sent to the same consolidator so that only one shipping box is generated per order. But since each tote represents SKUs for many orders, plus the items in an order are likely to be in multiple totes, a perfect assignment is not possible. The assignment objective then is to minimize Ship Boxes/Order over the shift while balancing consolidator utilization.

1.5.1.2 Develop a Mixed Integer Programming (MIP) model. The optimization model needs to make the decision on assigning picked totes to consolidators. Using the picking list and customer order list, a binary matrix with rows as totes and columns as orders is generated. The binary numbers in a tote-order matrix denote that if totes have SKU items of orders. The totes have more orders in same are more correlated. Our objective is to assign the correlated totes to the same consolidator, therefore, the consolidator can pack the SKUs from same order into one package for delivery. One of most important decision variables in this operation process is to decide which tote should be assigned to which consolidator, we call this tote assignment list. To avoid the situation that all totes in one planning horizon are assigned to one consolidator, we bring in the tote quantity balance as the consolidator capacity constraint. The first three research objectives relate to the Order Consolidation

Problem in fulfillment warehouses, while the fourth objective relates to Buy Online fulfill from Store problem.

1.5.2 Develop and Test Fast Heuristics for Solving the Order Consolidation Problem

The size of a MIP model, especially the binary variables and constraints, is the most discussed factor of the difficulty of a MIP problem. As problem size and complexity level of tote-order matrix increase, directly solving our MIP problem is not easy. The required computation time to verify the optimal solution could often become unbearable because of the enormous amount of integer variables involved.

1.5.2.1 Develop Fast Heuristics. The objective of the tote assignment list is to move the associated totes to the same consolidator. Imagine that the capacity of a small order consolidation zone is 20 totes and there are five consolidators working in the zone. We would like to organize all the 20 totes into five groups so that each group can be assigned to a different consolidator. Strategically, we would like that the totes in each group are as similar as possible. Moreover, two given totes having very different order patterns should not be placed in the same group. Our intention behind this operation strategy is to pack as many as possible SKU items for an order together and minimize the ship packages per order. We transform the tote assignment problem into the tote consolidation/clustering problem and developed the heuristic algorithm by using clustering techniques.

1.5.2.2 Implementation of the k-means and Hierarchical Clustering Methods.

During the process of developing the heuristic algorithm, we also applied the current existing clustering techniques on our data set. Cluster analysis has been widely used in many applications. Different clustering methods may generate different clusters on the same data set. Clustering methods can differ with respect to the partitioning level, whether or not clusters are mutually exclusive, the similarity measures used, etc. For our tote-order data sets, we adopt exclusive cluster separation, that is, each tote must assign to exactly one consolidator. And we use the distance-based similarity measures since distance-based methods can often take advantage of optimization techniques. Considering all aspects and requirements, we implement the k-means and hierarchical clustering methods on our problem.

1.5.3 Evaluate the Performance Behavior of the Fast Heuristics as a Function of Several Characteristics

The order consolidation is a dynamic process, items belong to different orders carrying by different totes come to the consolidation station continuously. Performance behavior of the heuristics is further studied as a function of following characteristics.

1.5.3.1 Size of Tote-Order Matrix. The size of the tote-order matrix is a critical factor to the complexity of the consolidation problem. We expand the data sets size to evaluate the robustness and performance of the fast heuristics with k-means and hierarchical clustering methods.

1.5.3.2 Multi-Item Order Complexity of Tote-Order Matrix. The number of items in orders contribute the complexity of the tote-order matrix. We design a set of order complexity levels to test the effectiveness of the heuristics.

1.5.3.3 Number of Consolidators. A variety of data sets are tested with different number of consolidators. A Maxmin ratio indexed in the range number of ordered items minus number of orders was used as a surrogate for solution performance.

1.5.3.4 Consolidation Tote Batch Window. The size of the tote batch window is a partitioning decision on the dynamic flow of totes. The tote-order matrix with dynamic design are generated to simulate the tote flow, then divided into smaller tote batches to apply the heuristics and analyze the results.

1.5.3.5 Twining Design. This is a preliminary design for the future research. The twinning design is the order similarity percentage in the tote-order matrix.

1.5.4 Back Stock Assignment and Picker Scheduling in a BOPS Store

Many retailers, particularly in the grocery business, have built their online sales strategy around a BOPS operation. This research solves two BOFS related problems: (i) Back stock strategy: Assignment of items located in the back stock and (ii) Picker scheduling: Effect of numbers of picker and work hours. For both problems, we assume a continuous flow of incoming orders and the objective is fulfillment time and labor cost minimization. We model the store inventory dispersion, order arrival process and order picking process in a BOPS retailer. As online orders continuously entering the system, an arrival time is stamped on each order. An order can consist of one or more items. When the picker schedule starts, a picker goes to collect items of one order from the online order pick pack area and come back after picking them all. Then a fulfilled time is stamped to this order. The difference between these two timestamps is the order fulfillment time, and it decided by the pick travel distance and picker schedule. We formulate the following problems to, first, minimize the order picking time and, second, minimize the order fulfillment cost. Simulation of online grocery order picking is used to compare several decision methods.

1.5.4.1 BOPS Store Stocking Layout Problem. The pick travel distance is determined by the stocking layout. The forward stock is the common retail section of a physical store, this section is arranged for customers browsing displayed products and cannot be physically rearranged. The back stock is the fast pick area but the shelving space capacitated, only a limited number of SKUs are selectively located here. The problem is which SKU items should be back stocked. We explicit the item location model and back-stock decision strategy for this problem.

1.5.4.2 BOPS Picker Scheduling Problem. The picker labor cost is the primary direct cost of BOP/FS. Long picker schedule will increase the labor cost, while short picker schedule will increase the order waiting cost. The picker start time also decides the order fulfilled time. In the picker scheduling problem, we develop the picker schedule optimization model to minimize the BOFS fulfillment cost.

CHAPTER 2

LITERATURE REVIEW

2.1 Internet Fulfillment Warehouses

Internet Fulfillment Warehouses (IFWs) present a new operational model in the design and control of warehouses. Structurally different, they are a key entity in transforming the global retail economy (Onal et al., 2018). We focus the research literature on the fulfillment side which focuses on the storage of products and their shipment to customers once an order is submitted through the web store.

Agatz et al. (2008) review internet fulfillment and multi-channel distribution and conclude that companies must embrace novel strategies to succeed. Commenting on the warehousing differences of online retail fulfillment, Bakker et al. (2016) highlight the much higher order frequency and the much smaller pick quantities. They advise that this requires material handling considerations quite different from traditional warehousing. Lee and Whang (2001) argue that fulfillment speed or immediacy is critical to winning in online retail. Acimovic and Graves (2015) found that quick fulfillment warehouses are unique to online retail, and involve picking, packing, and shipping in rapid succession. Gong et al. (2010) and Gong and Koster (2008) observe that order fulfillment is the most expensive and critical operation for companies engaged in e-commerce and immediacy is the primary challenge in building an efficient IFW. They analyzed the performance of a real-time picking and sorting systems in a general parallel-aisle warehouse. They recommend the use of a dynamic picking system in which a worker picks orders that arrive in real time during the picking operations. Other researchers have investigated different aisle layout strategies for faster fulfillment. Petersen and Aase (2017) examine a combination of cross aisles and storage policies on order picking times. They found that if across-aisle storage is used, then any cross-aisle configuration is sufficient to reduce picker travel. Tarn et al. (2003) observe that IFWs operate in a dynamic environment in which product and information are highly synchronized to achieve unprecedented levels of customer service. They note that traditional distribution systems established for retailers are not designed to accommodate the needs of individual customers with a large variety of small orders. Gunasekaran and Ngai (2004) identify information transparency as a key requirement of e-commerce supply chains, labelling it as the logistics information network enterprise. Hubner et al. (2016) surveyed the distribution operations of Omni-channel retailers, including internet fulfillment. They identify optimizing modes of delivery, increasing delivery speed, and inventory transparency as the key factors in achieving fulfillment excellence. Burns and Towers (2014) use the fast-changing fashion industry to highlight the need for manufacturers' to create new production planning and control systems if they are to succeed in an Omni-channel environment. Li et al. (2016) analyzed the order pick function of online retailers' warehouse operations and found four optimization strategies: warehouse layout, position allocation, order batching, and picker routing. Developing a solution for any of these strategies, will require an understanding of the underlying IFW operations, which this research provides.

Pan et al. (2017) and Montreuil (2017) introduced the novel concept of the physical internet. They define it as a hyper connected logistics system which improves by an order of magnitude, the efficiency by which physical objects are moved, deployed, realized, supplied, designed, and used. At the center of this network they identify a fulfillment center, where a digital customer order is converted into a physical delivery package. This research provides a detailed view of a fulfillment center and contributes to the knowledge of how the physical internet is being operationalized and built. The flows and product arrangements presented here, show that traditional warehousing methods, which are well documented in multiple textbooks (Bartholdi and Hackman, 2014; Tompkins et al. 2010), are evolving into new physical designs and operational control models that are better suited to meet fast fulfillment objectives.

2.1.1 IFW Differentiators

Onal et al. (2017) investigates the warehousing operations of internet retailers, based on observational studies of internet IFW operations at a leading internet retailer. The investigations find that traditional warehousing methods are being replaced by new methods which better leverage information technology and efficiently serve the new internet retail driven supply chain economy. They identified six key differentiators between an IFW and traditional warehouses or fulfillment centers. Together these differentiators provide the platform for the most successful online retailers to achieve their fast fulfillment objectives. These differentiators also show that a traditional warehouse cannot be easily transformed to an IFW, and significant structural and operational changes are required. The six differentiators are summarized here: (i) explosive storage policy (ii) very large number of beehive storage locations (iii) bins with commingled SKUs (iv) immediate order fulfillment (v) short picking routes with single unit picks and (vi) high transaction volumes with total digital control.

2.1.2 Fulfilment Time and Online Orders

The fundamental premise of this study is that faster fulfilment is a key driver in motivating customers to switch from a physical store visit to an online delivery order. Further, it is also a factor in selecting between online retailers. Several studies have discussed this relationship and a recent survey (Wall Street Journal, 2016) found that online shoppers want faster delivery with the maximum wait time dropping very year. Griffis et al. (2012) found that excellent order fulfilment is instrumental in generating referrals for the online retailer, even after factoring in product quality.

Several scales for evaluating service quality in electronic or online retailing have been proposed (Blut, 2016; Stiakakis and Georgiadis, 2009) and these all include fulfilment as a key factor. Though the emphasis in these scales is more on delivery reliability against a promised date, and less on the fulfilment time length. In a survey of online customers both Koufteros et al. (2014) and Jain et al. (2017) found that timeliness positively influenced customer satisfaction. Dholakia and Zhao (2010) also studied and compared customer evaluations of online purchases. They found that on-time delivery dominates customer satisfaction. Further, they note that weak fulfilment will not compensate for creative and vivid website designs. Meller (2015) quotes a recent survey of online buyers, which found that 65% want next day delivery and 24 said same-day delivery was important to them. They propose that faster fulfilment will allow retailers to expand their customer base by targeting the speed-sensitive segment. Specifically, they identify the order processing window, or time it takes to process an order in the fulfilment center, as a key determinant of success.

Bell et al. (2014) propose an information and fulfilment matrix to categorize Omni channel retailers. They note that fulfilment through package delivery is disadvantaged from the customer perspective by waiting time and delayed gratification. This then implies that the shorter the delivery fulfilment time then higher the likelihood a customer will switch from a physical store purchase to online, assuming equivalent pricing and quality. Further, when evaluating online retail choices, the faster fulfilment will be selected. In a comparison of offline and online retail channels, Lieber and Syverson (2012) describe fulfilment time as a delayed consumption which can be penalized by a discounted utility function. They propose that this delay can be quite significant when considering the interaction between a market's online and offline channels. Li et al. (2015) present a consumer utility model for online retailing, which includes the discount component rt. In their model, r measures the consumer's patience such that a smaller r implies more patience, and t is the fulfilment time. As the penalty increases, then as a consequence of the decreasing utility the consumer may choose a different retail option.

2.1.3 Likelihood of an Online Purchase

When multiple retail options are available then each will have its unique t, and the penalty function of Li et al. (2015) could be translated into the probability of a consumer selecting a specific online retailer. Figure 2.1 proposes that this probability be described by a non-linear decreasing function of t. In Figure 2.1, the function assumes that for same day delivery a maximum likelihood for online purchase is reached. Since a portion of customers will always demand immediate fulfilment, the maximum likelihood will be less than one. For an online retailer to be successful, it must therefore offer delivery times close to same day delivery. This is reflected in Amazon's progressively shorter fulfilment time targets: 2-day, next day, and now same day. Depending on the nature of the product and the associated consumer behavior, the waiting time disadvantage, indicated by r, could be steep or shallow as shown in Figure 2.1. For products with a steep disadvantage curve, such as grocery items, fulfilment must be within a day to ensure retail success. Bell et al. (2012) also identify several disadvantages of a physical store purchase, which could be modelled into a relationship describing the trade-off between physical and online purchase as a function of fulfilment time. Interestingly, Harris et al. (2017) found that the desire to avoid disadvantages maybe a stronger motivator for making the online or offline purchase.

2.1.4 Logistics Efficiency and Fulfilment Time

This research investigates whether the new logistics designs and approaches implemented by Amazon, lead to faster fulfilment times. Nguyen et al. (2016) presented a framework linking order fulfilment aspects with online consumer buying. Weapply

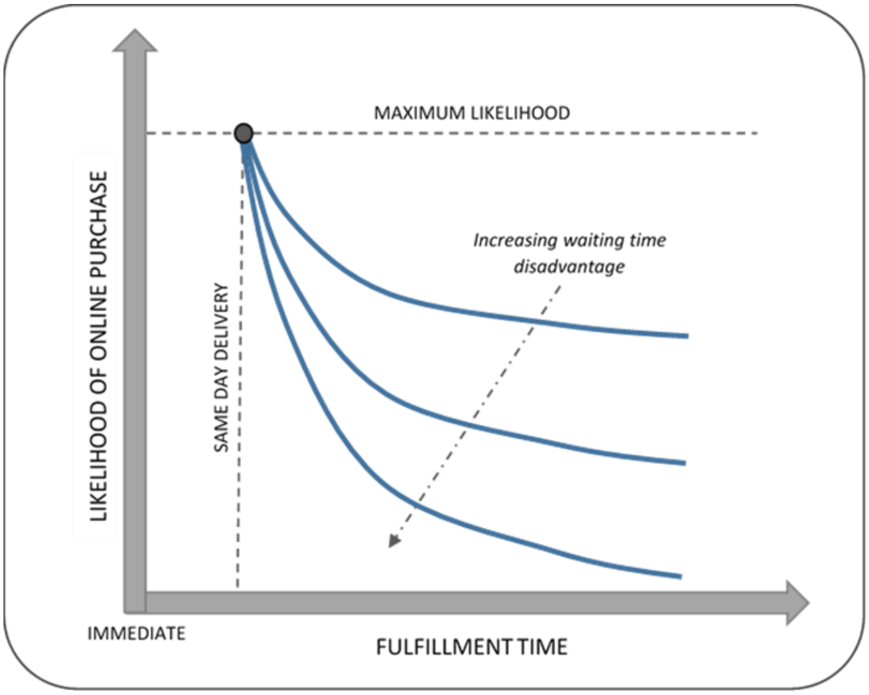


Figure 2.1 Online purchase likelihood and fulfilment time.

and extend their framework to present an expanded view (Figure 2.2) of the key functions in online order fulfilment and the associated performance drivers. This is based on facility visits and analytical reviews of Amazon fulfilment centers (Onal et al. 2017a), where the process flows, and facility design are radically different from that of any other online retailer (Onal et al. 2017b). As shown in Figure 2.2 the process is initiated by the receipt of a customer order, and a key performance driver is inventory management. The majority of leading online retailers will only accept orders if there is a fillable inventory and identify a stock out immediately to the customer. The order reject rate due to stock outs is then determined by the inventory policy. In this study, all the orders tracked were fillable, and the results therefore independent of inventory stocking policy. Where the inventory is stocked, which warehouse and where in the warehouse, though would be integral to the warehousing logistics function. The next two steps, pick and pack, and ship and transport, represent the key competitive advantage of Amazon. The performance driver here is warehousing logistics, which allows Amazon to achieve its quick fulfilment goals. Figure 2.2 highlights several innovative features in Amazon's fulfilment infrastructure that differentiate it from the competition. In particular, an explosive storage policy and bins with commingled SKUs are unique features. Most other retailers have designed their online fulfilment logistics, around existing warehouses and stores using more classical approaches.

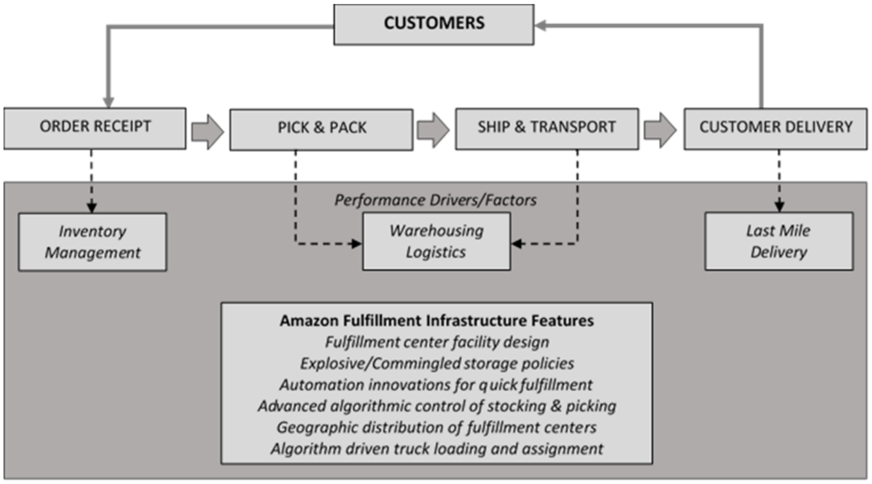

CUSTOMERS
ORDER RECEIPT
PICK & PACK
SHIP & TRANSPORT
CUSTOMER DELIVERY
Performance Drivers/Factors
Inventory Management
Warehousing Logistics
Last Mile Delivery
Amazon Fulfillment Infrastructure Features
Fulfillment center facility design
Explosive/Commingled storage policies
Automation innovations for quick fulfillment
Advanced algorithmic control of stocking & picking
Geographic distribution of fulfillment centers
Algorithm driven truck loading and assignment

Figure 2.2 Online fulfilment process and performance drivers.

Last mile delivery is also a key factor in faster fulfilment and many online retailers have established fast delivery arrangements with third-party delivery services such as FedEx, UPS, and the US Postal Service. This study is unable to differentiate the efficiency advantages of warehousing logistics and last mile delivery. But a reasonable assumption is that the efficiencies of third-party delivery services are available to all online retailers. The primary research question then is to confirm and quantify the online fulfilment time advantage that Amazon has achieved as a result of its warehousing logistics infrastructure.

2.2 Packing Operations

Operations in Amazon can be divided into three sub-operations: (i) Inbound shipment, (ii) Picking and Packing, and (iii) Outbound shipment.

Luna (2015) explores cycle time reductions and throughput adjustments required to reduce the Service Level Agreement (SLA) at one of Amazon’s Fulfillment Centers. The author goes into the details of the multiple activities for outbound operations. The orders that contain only one item called singles, and the orders that contain more than one item called multis. Based on order type, the totes are routed to a singles packing area or to a multis sorting area. Totes that contains all the items from the group of orders are called batches. The conveyor system routes all totes in a batch to the same location. The next activity is called tote wrangling; and it consists of associates with handheld scanners pulling totes from the conveyors and placing them on batch carts, effectively regrouping all the items from all the orders in one batch. A batch cart has numerous items from numerous orders on different totes. The next activity is called rebin, the items from the different orders need to be segregated from multiple totes into each individual order, and it consists of an associate at a rebin computer station sorting items from the numerous totes into bins on a cart.

In recent years, new types of facilities have also emerged, such as sortation centers where items from different fulfillment centers are consolidated to reduce the number of shipments. In general, sortation centers are smaller operations that can be located besides, adjacent to, or nearby larger fulfilment centers. The primary role of the sortation center is to aggregate shipments from one or more fulfilment centers for delivery into a defined regional grouping of zip codes typically belonging to a nearby set of populated urban areas (2014).

2.2.1 Operational Objectives

In online retailing, the main objective is optimizing the order fulfillment time while minimizing the relative supply chain costs. The primary objective of the fulfillment decision is to minimize the outbound shipping cost from fulfillment centers to customer.

Chen (2017) studies the decisions of inventory placement and inventory replenishment in online retail. The author used a mixed-integer program to formulate placement decision and dynamic program to formulate replenishment decision for a single item. The objective of the inventory placement modeling is to minimize the sum of outbound shipping and fixed costs for all the items, while satisfying demand and capacity constraints. The objective of the inventory replenishment modeling is to minimize the long-term average of outbound shipping, stockout and holding costs.

Acimovic (2012) and Acimovic and Graves (2015) focus on how an online retailer should choose the specific facilities from which to fulfill each order in order to minimize average outbound shipping costs. The online retailer decides from where items will ship, by what shipping method, and how or whether multiple-item orders will be broken up into multiple shipments. Only outbound shipping costs are considered: in general, it is more expensive to ship an item by air than by ground, and it is more expensive to ship a multi-item order in multiple packages than to ship it in a single package from a single fulfillment center. They develop a heuristic that makes fulfillment decisions by minimizing the immediate outbound shipping cost plus an estimate of future expected outbound shipping costs.

Xu (2005) and Xu et al. (2009) focus on the entire network of warehouses and customers. When a customer places an order on an e-tailer's website, the e-tailer assigns the order to one or more warehouses mainly based on the transportation cost of shipping the order from the warehouse(s) to the customer location and on the current warehouse inventory availability. They show the real-time decision is necessarily myopic because the e-tailer does not anticipate any future customer orders or inventory replenishment. Reducing the number of shipments is a very good proxy for minimizing the transportation costs in the e-tailing setting. The author construct near-optimal heuristics for the re-assignment for a large set of customer orders with the objective to minimize the total number of shipments.

2.2.2 Bin Packing Problem and Algorithms

The definition of the canonical bin packing problem is as follows: the weight capacity of the bins is fixed, and the goal is to pack items into the weight-constrained bins with the objective of minimizing the total number of bins. There are many variations of this problem. They can be classified by different criteria. The number of dimensions for the bins and items can be 1D, 2D or 3D. Depending on whether the algorithm can see all the items beforehand, there are on-line and off-line bin packing problems. The number of different candidate bin types divides the bin packing problems into the single sized bin packing and the variable sized bin packing.

Hall et al. (1988) study the classic problem of bin packing in one dimension. They adapt a variety of heuristic methods for generating feasible solutions, and the results are then used to generate confidence intervals for the (in practice unknown) value of the optimal solution. They selected ten bin packing heuristics which have been commonly addressed in the literature, including on-line heuristics and off-line heuristics. On-line heuristics pack items into bins as the items are generated. Off-line heuristics require that the number of items and their sizes be known before packing begins.

Modified bin-packing problem (Brusco et al. 1997; Rao and Iyengar 1994), is modeled using a fixed number of bins with no weight capacity and the objective is to pack items into the bins such that the sums of the item weights within each bin are as evenly distributed as possible, they used a squared deviation from target as the objective function. This is an especially important application in the psychometric literature pertains to splitting of a set of test items to create distinct subtests, each containing the same number of items, such that the maximum sum of item weights across all bins is minimized. Brusco et al. (2012) present a mixed zero-one integer linear programming (MZOILP) formulation of the one-dimensional minimax bin-packing problem and develop an approximate procedure for its solution that is based on the simulated annealing algorithm.

The other variant application of bin packing is efficient management of data center resource. The efficient management of a data-center involves minimizing energy costs while ensuring service quality. In one context, servers can be viewed as bins and virtual machines as items. The assignment of virtual machines on servers and how these servers are utilized has a huge impact on the energy consumption. Cambazarda et al. (2015) focus on a bin packing problem where linear costs are associated to the use of bins to model the energy consumption. They study lower bounds based on linear programming and extend the bin packing global constraint with cost information. Another strategy for reducing the energy consumption is workload consolidation that usually achieved by allocating multiple tasks on the same physical machine. Armant et al. (2017) leverage semi-online optimization techniques in which workload allocations must be made without full knowledge of future demands. They formalize the workload consolidation problem as a semi-online bin-packing problem whereby each bin maps to a machine and each item maps to a task.

Another paper introduces a deep learning approach to solve the 1D variable sized bin packing problem (Mao et al. 2017). They first define the optimization heuristics space for this particular bin packing problem. Then, they model a large neural network to predict the optimal strategy for each bin packing instance.

2.3 Clustering Algorithms in Order Fulfillment

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Cluster analysis has been widely used in many applications such as business intelligence, image pattern recognition, Web search, biology, and security. In business intelligence, clustering can be used to organize a large number of customers into groups, where customers within a group share strong similar characteristics. This facilitates the development of business strategies for enhanced customer relationship management.

In Vinod's paper (1969), the author points out that the problem of grouping, where a larger number of elements n are combined into m mutually exclusive groups (m¡n) should be recognized as a problem in Integer Programming. He constructs two mathematical formulations of the grouping problem. Both formulations are based on integer variables that can take values 0 or 1 only. The first formulation uses a constraint set similar to that of the warehouse location problems. The second formulation discusses the problem of minimization of the within-group sum of squares.

Cluster analysis involves the problem of optimal partitioning of a given set of entities into a pre-assigned number of mutually exclusive and exhaustive clusters. In Rao's paper (1971), this problem is formulated in two different ways with the distance function (a) of minimizing the within groups sums of squares and (b) minimizing the maximum distance within groups. When the entities can be represented as points on the real line and the criterion is to minimize the within groups sums of squares, an efficient dynamic programming algorithm was obtained. With the same criterion, when the entities can be represented as points in a multidimensional Euclidian space, an integer linear programming formulation was given.

The basic premise is to utilize a distance or dissimilarity matrix to group items together based upon one or more attributes. The general methodology requires (1) the determination of a distance matrix and (2) the clustering of items one by one in a bottom-up approach or decomposing the entire set of items into two groups successively in a top down approach. Klein (1991) formulate a mixed-integer programming model for optimal clustering based upon scaled distance measures to account for this total group interaction.

The purpose of Hansen and Jaumard's paper (1997) is to review the mathematical programming approach to cluster analysis. A survey is given from a mathematical programming viewpoint. Steps of a clustering study, types of clustering and criteria are discussed. Then algorithms for hierarchical, partitioning, sequential, and additive clustering are studied. Emphasis is on solution methods, i.e., dynamic programming, graph theoretical algorithms, branch-and bound, cutting planes, column generation and heuristics.

One application of cluster analysis is in cellular manufacturing (CM). CM is a special application of group technology (GT) which is used to cluster parts into families and machines into cells for efficient production. Berardi et al. (1999) investigate the effect of the alternative starting part family/machine cell clusters on the solution of the mathematical programming model.

In graph theory, given a graph G = (V,E), where V is the vertex set and E is the edge set, cluster analysis refers to finding a partition of V into disjoint groups called clusters (or communities) such that vertices in the same cluster are densely connected to each other and less connected to those in other clusters. In order to identify clusters in a graph, clustering can be formulated in mathematical programming with an objective function to optimize. With given nonnegative edge weights, Hassin, R. and Rubinstein, S. (2006) describe an approximation algorithm for maximizing the sum of weights of edges whose two ends belong to the same cluster. Another clustering measure is modularity density, the optimization problem is Modularity Density Maximization (MDM) problem. Costa et al. (2017) derive several exact Mixed-Integer Linear Programming (MILP) reformulations of auxiliary binary NLP problems, and obtain complete MILP formulations of MDM.

2.4 Optimization Platform

2.4.1 Modeling Language

Algebraic modeling languages are sophisticated software packages that provide a key link between an analyst's mathematical conception of an optimization model and the complex algorithmic routines that seek out optimal solutions (Robert Fourer, 2013). By allowing models to be described in the high-level, symbolic way that people think of them, while automating the translation to and from the quite different low-level forms required by algorithms, algebraic modeling languages greatly reduce the effort and increase the reliability of formulation and analysis. They have thus played an essential role in the spread of optimization to all aspects to OR/MS and to many allied disciplines.

AMPL is a language for algebraic modeling and mathematical programming: a computer-readable language for expressing optimization problems such as linear programming in algebraic notation (Fourer, Robert and Brian W. Kernighan, 2002). Optimization problems arise in many contexts (David M. Gay). Sometimes finding a good formulation takes considerable effort. A modeling language, such as AMPL, facilitates experimenting with formulations and simplifies using suitable solvers to solve the resulting optimization problems. AMPL lets one use notation close to familiar mathematical notation to state variables, objectives, and constraints and the sets and parameters that may be involved. AMPL does some problem transformations and makes relevant problem information available to solvers. The AMPL command language permits computing and displaying information about problem details and solutions returned by solvers. It also lets one modify problem formulations and solve sequences of problems. AMPL addresses both continuous and discrete optimization problems and offers some constraint programming facilities for the latter. More generally, AMPL permits stating and solving problems with complementarity constraints. For continuous problems, AMPL makes first and second derivatives available via automatic differentiation. The freely available AMPL/solver interface library (ASL) facilitates interfacing with solvers. This paper gives an overview of AMPL and its interaction with solvers and discusses some problem transformations and implementation techniques. It also looks forward to possible enhancements to AMPL.

Practical large-scale mathematical programming involves more than just the application of an algorithm to minimize or maximize an objective function (Robert Fourer et al. 1990) Before any optimizing routine can be invoked, considerable effort must be expended to formulate the underlying model and to generate the requisite computational data structures. AMPL is a new language designed to make these steps easier and less error prone. AMPL closely resembles the symbolic algebraic notation that many modelers use to describe mathematical programs, yet it is regular and formal enough to be processed by a computer system; it is particularly notable for the generality of its syntax and for the variety of its indexing operations. We have implemented a translator that takes as input a linear AMPL model and associated data and produces output suitable for standard linear programming optimizers. Both the language and the translator admit straightforward extensions to more general mathematical programs that incorporate nonlinear expressions or discrete variables.

2.4.2 Solvers

AMPL Optimization also supports the free solver programs of CPLEX, Gurobi, and Xpress for academic use. All academic versions allow full use of machine resources, with problem sizes limited only by the memory, storage, and processors available. The report of David M. Gay (1993) tells how to make solvers work with AMPL's solve command. It describes an interface library, amplsolver.a, whose source is available from netlib as individual files, as gzip-compressed files, or in a single tar file. Examples include programs for listing LPs, automatic conversion to the LP dual (shell-script as solver), solvers for various nonlinear problems (with first and sometimes second derivatives computed by automatic differentiation), and getting C or Fortran 77 for non-linear constraints, objectives and their first derivatives. Drivers for various well known linear, mixed-integer, and nonlinear solvers provide more examples.

2.4.3 Network-Enabled Optimization System (NEOS) Server

The NEOS Server is a free internet-based service for solving numerical optimization problems. It provides access to more than 60 state-of-the-art solvers in more than a dozen optimization categories and offers a variety of interfaces for accessing the solvers to enable jobs run on distributed high-performance machines. Czyzyk, J. et al. (1998) discusses the design and implementation of the NEOS Server. Dolan, E. (2001) discusses the implementation of the server and its use in detail. Gropp, W. and Moré, J. J. (1997) discusses the NEOS Server as a problem-solving environment that simplifies the formulation of optimization problems and the access to computational resources.

2.4.4 SolverStudio

SolverStudio is an add-in for Excel that allows you to build and solve optimization models in Excel using many optimization modeling languages. SolverStudio allows models built using AMPL to be solved using the NEOS server. Mason AJ (2013) provides a basic introduction to SolverStudio. SolverStudio is written in VBA, and C using Visual Studio 2010 Professional. It uses the Microsoft VSTO (Visual Studio

Tools for Office) system running on .Net 4 to manage the integration with Excel. It includes IronPython as its embedded Python engine.

CHAPTER 3

THE ORDER CONSOLIDATION PROBLEM

3.1 The Order Picking and Consolidation Process

Figure 3.1 flowcharts the F-Warehouses order picking and consolidation process. The order receipt process is data driven with orders arriving continuously, which are then immediately updated to the customer order list. A customer order may contain more than one SKU item, in which case each SKU generates a separate record with a common order number. Whenever a picker becomes free the F-Warehouses control logic uses the customer order list and the inventory state records to generate an order picking list. The picker first links an empty tote to the assigned list, and then follows the sequential picks to fill the tote.

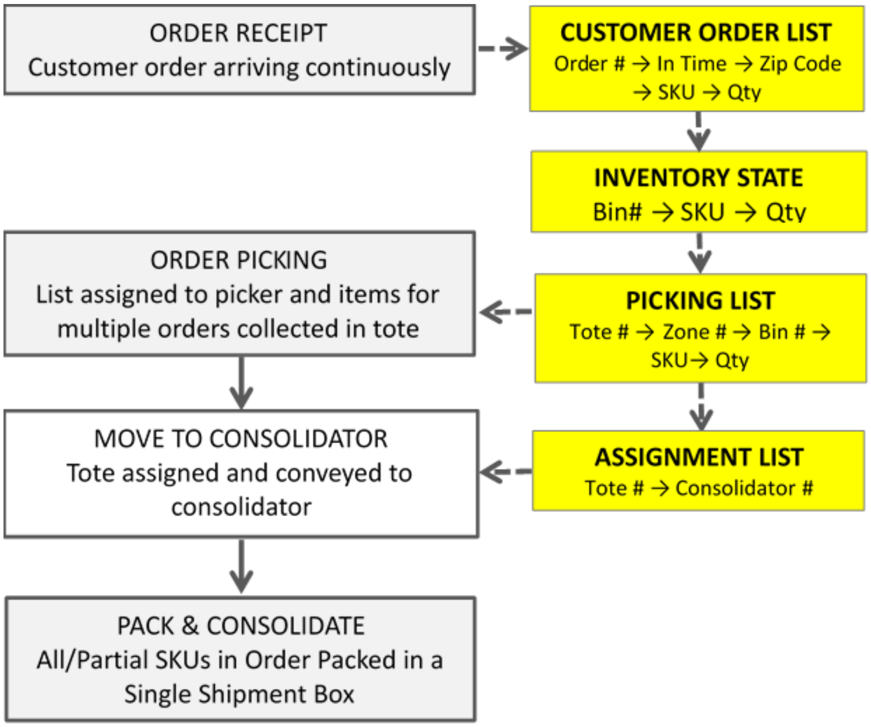

ORDER RECEIPT
Customer order arriving continuously
CUSTOMER ORDER LIST
Order # → In Time → Zip Code → SKU → Qty
INVENTORY STATE
Bin# → SKU → Qty
ORDER PICKING
List assigned to picker and items for multiple orders collected in tote
PICKING LIST
Tote # → Zone # → Bin # → SKU→ Qty
MOVE TO CONSOLIDATOR
Tote assigned and conveyed to consolidator
ASSIGNMENT LIST
Tote # → Consolidator #
PACK & CONSOLIDATE
All/Partial SKUs in Order Packed in a Single Shipment Box

Figure 3.1 Order picking and consolidation process.

At picking stage, one picker may work on multiple orders, and the items from one order may be picked by one picker or by multiple pickers. Therefore, each item in the order could be placed into different totes throughout different locations in the F-Warehouses. In other words, there's small probability that the items form one order being picked simultaneously at the same location. This also can be explained by the differentiators Onal et al. (2017) identified between an F-Warehouses and traditional warehouses or fulfillment centers. Below listed the differentiators of them.

Explosive Storage Policy - An incoming bulk SKU is exploded into multiple storage lots such that no lot contains more than 10% of the received quantity, the lots are then stored in random locations anywhere in the warehouse without preset restrictions.

Very Large Number of Beehive Storage Locations - Storage is organized into small bins as opposed to large bulk holding spaces. The entire IFW is organized into racks that are divided into many small bins in a sort of beehive pattern. A million square-foot IFW could therefore have several million bins. A similar sized traditional warehouse may have only 10,000 locations. This is the most apparent physical difference of an IFW.

Bins with Commingled SKUs - One of the most radical differentiators of an IFW, is that multiple SKUs are simultaneously stored in the same bin.

Short Picking Routes with Single Unit Picks - Order picking efficiency is a key decision in warehouse operations, and the pick list decision problem is focused primarily on travel time minimization. In an IFW most orders are for only a few units and in most cases for only a single unit. A pick list therefore retrieves several different items within a short pick zone. This is made possible by the explosive storage policy and beehive storage.

Since a customer order may consist of more than one SKU, then ideally all SKUs should be shipped together. But each SKU could be picked in a different zone and be in a different tote. Consolidation is the inverse of explosion and the objective of the assignment list is to move the associated totes to the same consolidator. Note that there is no perfect solution and at the end of the day there will be more boxes shipped than orders. Assignment is constrained by the tote composition, pick time for each SKU and the processing capacity of each consolidator station. Using the picking list and customer order list, the F-Warehouses control logic generates a tote assignment list. Each consolidator station is manned by a single worker. Physically, the worker is surrounded by racks of totes filled with picked products. During the pack and consolidate activity, the consolidator follows the display instructions to setup an order labelled shipment box and fills it with specific SKUs picked from one or more totes. When done, the box is packed and forwarded to shipping.

3.2 The Tote Assignment Problem

In a high-volume F-Warehouse, hundreds of picked totes are generated every hour and the conveying system can direct a tote to any one of the available consolidators. The consolidator is then instructed to pick SKU items from a specific tote to create a shipping box for a customer order. In the ideal case, all SKUs for the order are sent to the same consolidator so that only one shipping box is generated per order. But since each tote represents SKUs for many orders, plus the items in an order are likely to be in multiple totes a perfect assignment is not possible. The assignment objective then is to minimize Ship Boxes/Order over the shift while balancing consolidator utilization. Then an optimization model would make the decision: picked tote is assigned to which consolidator.

Figure 3.2 illustrates the problem and shows the case where a single order generates three shipping boxes. Each consolidator station is also limited by the number of totes staged for shipping. This would be a constraint in the assignment problem, and an extended model would need to consider the tote queuing delays.

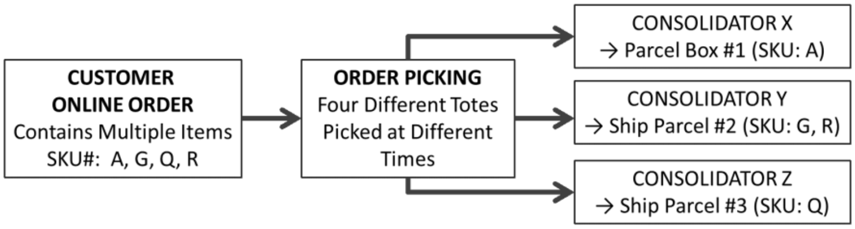


Figure 3.2 Tote assignment problem.

3.3 Problem Formulation

We develop a Mixed Integer Programming (MIP) model for the tote assignment problem. Using the picking list and customer order list, a binary matrix with rows as totes and columns as orders is generated. The binary numbers in this tote-order matrix denote that if totes have SKUs from each order. The totes have more SKUs from same orders are more correlated. Our objective is to assign the correlated totes to same consolidator, therefore the consolidator can pack the SKUs from same order into one package for delivery. One of most important decision variables in this operation process is to decide which tote should be assigned to which consolidator, we call this tote assignment list. In order to avoiding the situation that all totes in one planning horizon are assigned to one consolidator, we bring in the tote quantity balance as the consolidator capacity constraint.

Model Parameters:

K: Set of consolidators, consolidator $c \in$ K = {1…k}.

M: Set of orders, order $r \in$ M = {1…m}.

N: Set of totes, tote $i \in$ N = {1…n}.

$a_{i,r}$: Binary value of the tote-order matrix, for all r ∈ M, i ∈ N, means tote i has SKUs from order r if is 1, and 0 otherwise.

Decision Variable:

$\mathrm{x}_{\mathrm{i,c}}$: Binary variable, $\mathrm{x}_{\mathrm{i,c}} = 1$ if tote i is assigned to consolidator c, and $\mathrm{x}_{\mathrm{i,c}} = 0$, otherwise, for all i ∈ N, c ∈ K.

Auxiliary variables:

$z_{c,r}$: Shipment packages that consolidator c packs for order r, for all c ∈ K, r ∈ M, means consolidator c has SKUs from order r and pack them for shipment if is 1, and 0 otherwise.

b_c: Tote quantity balance for number of totes are assigned to consolidator c, for all c ∈ K.

The problem can now be formulated as the following mixed integer programming (MIP) model.

Min $\sum_{c=1}^{k} \sum_{r=1}^{m} z_{c,r} + \beta \sum_{c=1}^{k} b_c$ (3.1)

s.t. $\sum_{c=1}^{k} x_{i,c} = 1$ for all i ∈ N (3.2)

$z_{c,r} \geq x_{i,c} a_{i,r}$ for all r ∈ M, i ∈ N, c ∈ K (3.3)

$b_c \geq \sum_{i=1}^{n} x_{i,c} - n/k$ for all c ∈ K (3.4)

$b_c \geq n/k - \sum_{i=1}^{n} x_{i,c}$ for all c ∈ K (3.5)

$x_{i,c} \in \{0,1\}, z_{c,r} \geq 0, b_c \geq 0$ for all r ∈ M, i ∈ N, c ∈ K (3.6)

In this model, the objective function (3.1) is to minimize the shipped packages with the tote quantity balance. There is an obvious solution that assign all totes to one consolidator to get the minimal number of shipment package. The tote quantity balance will bring the optimal value up to remove this extremely uneven totes assignment solution. β is the balance coefficient, we set it as 1 by default, and gradually increase it when tote quantity balance is insufficient to the extremely uneven totes assignment solution. Constraints (3.2) ensure that each tote can be assigned to only one consolidator. Constraints (3.3) ensure that if a tote i has order r, and this tote i is assigned to a consolidator c, then the consolidator c has to prepare a shipment package for the order r. Constraints (3.4) and (3.5) define the tote quantity balance is the difference between the average totes can be assigned to consolidators and the totes assigned to consolidators, where n is total number of totes, k is total number of consolidators. Constraints (3.6) restricts the domains of decision variables.

Measuring the difficulty of a MIP problem is a very difficult question. It depends on so many factors. The most discussed indicators of computational complexity for generic MIP in our search are:

Size of the formulation - number of variables and constraints, intuitively the number of binary variables, when solving the MIP through the branch-and-bound algorithm.

Tightness of the formulation - the gap between integer optimal value and the optimal value of linear relaxation.

The size of our MIP model depends on the numbers of consolidators, totes, and orders. The number of binary variables and constraints in each case are shown in Table 3.1.

For a given MIP problem, the knowledge of the polyhedral structure of the problem is crucial. And the attempt to reduce the CPU time of solving a given model has to be experimented empirically. Under different size of consolidators, totes, and orders, we have also generated three levels of tote-order matrix (easy, medium, hard), and have solved the models by using solver Gurobi. The CPU time in each case are shown in Table 3.2.

Table 3.1 Problem Sizes with Number of Variables and Constrains

Consolidators	*Totes*	*Orders*	*Binary variables*	*Total constraints*
5	**20**	**60**	100	6030
6	**30**	**80**	180	14442
7	**35**	**100**	245	24549
10	**50**	**140**	500	70070
15	**75**	**210**	1125	236355
20	**100**	**280**	2000	560140

Table 3.2 Problem Sizes with Computational Time (Seconds)

Consolidators	*Totes*	*Orders*	*Easy*	*Medium*	*Hard*
5	**20**	**60**	0	1	10
6	**30**	**80**	0	20	83
7	**35**	**100**	1	240	2532

We can see that as problem size and complexity level of tote-order matrix increase, directly solving our MIP problem becomes more difficult. The required computation time to verify the optimal solution could often become unbearable because of the enormous amount of integer variables involved. In many problems, adding variables helps strengthening the linear relaxations and hence, the bounds used at each node of the branch and bound algorithm. However, in some cases, adding valid inequalities just makes the model bigger and heavier at each node. We also have found that, in some cases, providing a strong feasible initial solution to a MIP has caused the solver to iterate endlessly trying to prove its optimality. The fact that modern solvers, like CPLEX or Gurobi have a really large number of parameters, heuristic presolvers and a gazillion things does not really help in the very least.

3.4 Fast Heuristic F#1

The objective of the tote assignment list is to move the associated totes to the same consolidator. Imagine that the capacity of a small order picking and consolidation zone is 20 totes, and we have five consolidators working in the zone. We would like to organize all the 20 totes into five groups so that each group can be assigned to a different consolidator. Strategically, we would like that the totes in each group are as similar as possible. Moreover, two given totes having very different order patterns should not be placed in the same group. Our intention behind this operation strategy is to pack as many as possible SKU items for an order together and minimize the ship packages per order. In order to accomplish this task, we transform the tote assignment problem into the tote clustering problem and developed the fast heuristic F#1 by using clustering techniques.

Clustering is the process of grouping a set of data objects into multiple groups or clusters so that objects within a cluster have high similarity but are very dissimilar to objects in other clusters. Dissimilarities and similarities are assessed based on the attribute values describing the objects and often involve distance measures.

3.4.1 Tote-Order Matrix and Dissimilarity Matrix

Data matrix is a data structure that have n objects (e.g., totes in our case) described by m attributes (orders in our case). The totes are $a_1 = (\, a_{1,1},\ a_{1,2}, \dots ,\ a_{1,m})$, $a_2 = (\, a_{2,1},\ a_{2,2}, \dots ,\ a_{2,m})$, and so on, where $a_{i,r}$ is the binary value for tote a_i of the *r*th order. For brevity, we hereafter refer to tote a_i as tote *i*.

Tote-order matrix (tote-by-order structure): This structure stores the n totes in the form of a relational table, or n-by-m matrix (n totes × m orders):

$$\begin{array}{c} a_{1,1} \dots a_{1,r} \dots a_{1,m} \\ \dots \quad \dots \quad \dots \quad \dots \quad \dots \\ a_{i,1} \dots a_{i,r} \dots a_{i,m} \\ \dots \quad \dots \quad \dots \quad \dots \quad \dots \\ a_{n,1} \dots a_{n,r} \dots a_{n,m} \end{array}$$

Each row corresponds to a tote. As part of our notation, we may use r to index through the m orders. A numerical example of tote-order matrix with 5 totes by 10 orders is shown below, binary values indicate that tote i has SKUs of order r if the matrix entry is 1, and 0 otherwise.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Dissimilarity matrix (tote-by-tote structure): This structure stores a collection of proximities that are available for all pairs of n totes. It is often represented by an n-by-n table:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \ldots & \ldots & \ldots & & \\ d(n,1) & d(n,2) & \ldots & \ldots & 0 \end{bmatrix}$$

where d(i, j) is the measured dissimilarity or "difference" between totes i and j. In general, d(i, j) is a non-negative number that is close to 0 when totes i and j are highly similar or "near" each other, and becomes larger the more they differ. Note that d(i, i) = 0; that is, the difference between an object and itself is 0. Furthermore, d(i, j) = d(j, i). (For readability, we do not show the d(j, i) entries; the matrix is symmetric.) A numerical example of dissimilarity matrix transformed from the above tote-order matrix example is shown below.

$$\begin{bmatrix} 0 & & & & \\ 0.857 & 0 & & & \\ 1 & 0.667 & 0 & & \\ 0.667 & 1 & 0.857 & 0 & \\ 0.857 & 0.857 & 0.857 & 0.857 & 0 \end{bmatrix}$$

Many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix. Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms.

3.4.2 Transform the Tote-Order Binary Matrix into the Dissimilarity Matrix

In the following 2 × 2 contingency table, where q is the number of orders that equal 1 for both totes i and j, u is the number of orders that equal 1 for tote i but equal 0 for tote j, s is the number of orders that equal 0 for tote i but equal 1 for tote j, and t is the number of orders that equal 0 for both tote i and j. The total number of orders is p, where p = q + u + s + t.

		Tote *j*		
		1	**0**	**sum**
Tote *i*	**1**	*q*	*u*	*q + u*
	0	*s*	*t*	*s + t*
	sum	*q + s*	*u + t*	*p*

$$d(i,j) = \frac{u+s}{q+u+s} \tag{3.7}$$

The coefficient d(i, j) is called the Jaccard coefficient and is popularly referenced in the literature. In the following numerical example, we continue using the above example of tote-order matrix and compute the Jaccard coefficient between tote 1 and tote 2 (first and second row).

		Tote 1		
		1	**0**	**sum**
Tote 2	**1**	1	3	4
	0	3	3	6
	sum	4	6	10

$$d(2,1) = \frac{3\ +\ 3}{1\ +\ 3\ +\ 3} = 0.857$$

3.4.3 Fast Heuristic F#1

When an algorithm uses the minimum distance to measure the distance between clusters, it is sometimes called a nearest-neighbor clustering algorithm. If we view the totes as nodes of a graph, with edges forming a path between the totes in a cluster, then the merging of two clusters, Ci and Cj, corresponds to adding an edge between the nearest pair of totes in Ci and Cj. Because edges linking clusters always go between distinct clusters, the resulting graph will generate a tree. Thus, an agglomerative hierarchical clustering algorithm that uses the minimum distance measure is also called a minimal spanning tree algorithm, where a spanning tree of a graph is a tree that connects all totes, and a minimal spanning tree is the one with the least sum of edge weights. In our application, we merge the two totes, ti and tj, with the minimum distance into a new bigger tote which will have all of the orders of the previous two totes, ti and tj, and set a boundary for tote quantity during the clustering process. Then iterate this step based on merged tote-order matrix until all of the original totes are assigned into clusters. The proposed algorithm is described by the steps:

Step 1 – According to Equation (3.7), transform the tote-order matrix (tote-by-order structure) into the dissimilarity matrix (tote-by-tote structure). As already mentioned above, many clustering and nearest-neighbor algorithms operate on a dissimilarity matrix. Data in the form of a data matrix can be transformed into a dissimilarity matrix before applying such algorithms.

Step 2 – According to the nearest-neighbor clustering algorithm, select the smallest dissimilarity value *min{*$d_{i,j}$*}* from the dissimilarity matrix (from step 1). The smallest dissimilarity value *min{*$d_{i,j}$*}* indicate the nearest pair of totes *i* and *j*. If there's tie, break it arbitrarily.

Step 3 – Merge the nearest pair of totes *i* and *j* with the *min{*$d_{i,j}$*}* (from step 2) into a new tote *i'* and update the merged tote-order matrix for the next iteration. All of the orders in the totes i and j go to the new tote i' after the merging. The logic can be formulated as shown below:

Min $\sum_{r=1}^{m} a_{i\prime,r}$

s.t. $a_{i\prime,r} \geq a_{i,r}$ for all r ∈ M

$a_{i\prime,r} \geq a_{j,r}$ for all r ∈ M

We still use the previous Example of Tote-Order Matrix and Dissimilarity Matrix to compute the Merged Tote-Order Matrix as a numerical example, and shown below:

$$\begin{bmatrix} 0 & & & & \\ 0.857 & 0 & & & \\ 1 & 0.667 & 0 & & \\ 0.667 & 1 & 0.857 & 0 & \\ 0.857 & 0.857 & 0.857 & 0.857 & 0 \end{bmatrix}$$

Example of Dissimilarity Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Example of Tote-Order Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Merged Tote-Order Matrix

Step 4 – Repeat step 1, 2, and 3 until all totes are assigned into clusters.

Similar to tote quantity balance in the MIP formulation, we set n/k as the boundary for number of totes in each cluster to avoid the situation that one cluster is assigned much more totes than other clusters. In other words, if the number of totes within a cluster is more than n/k, the fast heuristic F#1 will not put more totes into this cluster in the following iterations.

3.5 Clustering Methods in Data Mining

During the process of developing the fast heuristic F#1, we also applied the current existing clustering techniques on our data set. Cluster analysis is the process of partitioning a set of data objects into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. Cluster analysis has been widely used in many applications. Different clustering methods may generate different clusters on the same data set. Clustering methods can differ with respect to the partitioning level, whether or not clusters are mutually exclusive, the similarity measures used, etc. All these aspects with which clustering methods can be compared in different applications are listed below:

The partitioning criteria: In some methods, all the objects are partitioned so that no hierarchy exists among the clusters. Alternatively, other methods partition data objects hierarchically, where clusters can be formed at different levels.

Separation of clusters: Some methods partition data objects into mutually exclusive clusters. In some other situations, the clusters may not be exclusive, that is, a data object may belong to more than one cluster.

Similarity measure: Some methods determine the similarity between two objects by the distance between them. In other methods, the similarity may be defined by connectivity based on density or contiguity and may not rely on the absolute distance between two objects. Similarity measures play a fundamental role in the design of clustering methods.

Also, clustering algorithms have several requirements, such as:

Capability of clustering high-dimensionality data: Most clustering algorithms are good at handling low-dimensional data such as data sets involving only two or three dimensions. Finding clusters of data objects in a high-dimensional space is challenging, especially considering that such data can be very sparse

Constraint-based clustering: Real-world applications may need to perform clustering under various kinds of constraints. A challenging task is to find data groups with good clustering behavior that satisfy specified constraints.

Considering all above aspects and requirements, for our tote-order data sets, we adopt exclusive cluster separation, that is, each tote must assign to exactly one consolidator. And we use the distance-based similarity measures since distance-based methods can often take advantage of optimization techniques.

3.5.1 Partitioning Methods

Given a set of n objects, a partitioning method constructs k partitions of the data, where each partition represents a cluster and $k \leq n$, each cluster must contain at least one object. The basic partitioning methods typically adopt exclusive cluster separation, which means each object must belong to exactly one cluster. The number of clusters k is also given as background knowledge. This parameter is the starting point for partitioning methods. The partitioning method then uses an iterative relocation technique to improve the partitioning by moving objects from one group to another. Most partitioning methods are distance-based. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are “similar” to one another and “dissimilar” to objects in other clusters in terms of the data set attributes (Jiawei Han, Micheline Kamber, Jian Pei, Data Mining Concepts and Techniques).

Partitioning is the simplest and most fundamental version of cluster analysis, however, achieving global optimality in partitioning-based clustering is often computationally prohibitive, potentially requiring an exhaustive enumeration of all the possible partitions. Instead, most applications adopt popular heuristic methods, such as greedy approaches like the k-means and the k-modes algorithms, which progressively improve the clustering quality and approach a local optimum. In the following, we apply these two most well-known and commonly used partitioning methods on our tote assignment problem.

The k-means algorithm is a centroid-based partitioning technique that uses a centroid to represent a cluster. Conceptually, the centroid of a cluster is its center point. The centroid can be defined in various ways such as by the mean of the objects assigned to the cluster (Jiawei Han, Micheline Kamber, Jian Pei, Data Mining Concepts and Techniques). Given a set of objects and an integer number k ($\leq n$), the k-means algorithm searches for a partition of A into k clusters that minimizes the within groups sum of squared errors (WGSS).

In this application of k-means algorithm on our tote assignment problem, we use same notation as in above MIP formulation and heuristic F#1 (3.3 & 3.4) to keep unification. The tote-order data set A (n totes × m orders) can be considered as a set of totes $A = \{a_1, a_2, \ldots, a_n\}$, and each tote $a_i = (\ a_{i,1},\ a_{i,2}, \ldots\ ,\ a_{i,m})$ has exactly m attribute values to indicate if it contain item of an order. With the number of consolidators k ($\leq n$), the k-means algorithm can be formulated as the following mathematical program problem P:

Min $P(X, Q) = \sum_{c=1}^{k} \sum_{i=1}^{n} x_{i,c}\ d(a_i, q_c)$ (3.8)

s.t. $\sum_{c=1}^{k} x_{i,c} = 1 \quad 1 \leq i \leq n$ (3.9)

$x_{i,c} \in \{0,1\} \quad 1 \leq i \leq n, 1 \leq c \leq k$ (3.10)

where X is an n × k partition matrix, $Q = \{q_1, q_2, \ldots, q_k\}$ is a set of cluster means, Qc is the cluster of totes assigned to consolidator c, qc is the centroid of Qc. The difference between a tote ai∈Qc and qc is measured by d(ai, qc), which is the Euclidean distance between two objects.

The k-means procedure is summarized in the following.

```
Input:
k: the number of consolidators,
A: tote-order data set containing n totes.
Output: A set of k clusters.
Method:
1) arbitrarily choose k objects from A as the initial cluster centers;
2) repeat
3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
4) update the cluster means, that is, calculate the mean value of the objects for each cluster;
5) until no change;
```

The k-means method is not guaranteed to converge to the global optimum and often terminates at a local optimum. The results may depend on the initial random selection of cluster centers. To obtain good results in practice, it is common to run the k-means algorithm multiple times with different initial cluster centers. The problem of applying k-means algorithm on the tote-order data set is that every tote is binary object because it only has binary values. The k-means method can be applied only when the mean of a set of objects is defined. So we consider binary objects as numeric and calculate the means when applying the k-means method. For data with nominal attributes involved, most of the literatures introduce the k-mode algorithm to make the formulation of problem P also valid.

The k-modes method is a variant of k-means, which extends the k-means paradigm to cluster nominal data by replacing the means of clusters with modes. It uses new dissimilarity measures to deal with nominal objects and a frequency-based method to update modes of clusters (Jiawei Han, Micheline Kamber, Jian Pei, Data Mining Concepts and Techniques). Huang (1998) made the following modification to the k-means algorithm:

1. using a simple matching dissimilarity measure for categorical objects,
2. replacing means of clusters by modes, and
3. using a frequency-based method to find the modes

In our tote assignment problem, a_i and a_j are two totes described by m binary attributes. The dissimilarity measure between a_i and a_j can be defined by the total mismatches of the corresponding attribute categories of the two objects. The smaller the number of mismatches is, the more similar the two totes. Formally,

$$d(a_i, a_j) = \sum_{r=1}^{m} \delta(a_{i,r}, a_{j,r}) \tag{3.11}$$

Where

$$\delta(a_{i,r}, a_{j,r}) = \begin{cases} 0 & (a_{i,r} = a_{j,r}) \\ 1 & (a_{i,r} \neq a_{j,r}) \end{cases} \tag{3.12}$$

The clustering process minimizes the following objective function,

$$P(X, Q) = \sum_{c=1}^{k} \sum_{i=1}^{n} \sum_{r=1}^{m} x_{i,c}\, \delta(a_{i,r}, q_{c,r}) \tag{3.13}$$

In k-modes clustering, the cluster centers are represented by the vectors of modes of categorical attributes. However, the biggest problem of applying k-modes algorithm in our tote-order data set is that each object (tote) using binary value to represent if it has attribute (order) and attributes of 1 are sparse in the data set. In this case, all of the cluster centers will be the vectors of 0, and the k-modes algorithm cannot continue. For the completeness of this document, we still keep the k-modes algorithm steps as following. To cluster a categorical data set into k clusters, Huang (1998) summarize the k-modes clustering process consists of the steps:

1. Select k initial modes, one for each cluster.

2. Allocate an object to the cluster whose mode is the nearest to it according to (3.11). Update the mode of the cluster after each allocation.

3. After all objects have been allocated to clusters, retest the dissimilarity of objects against the current modes. If an object is found such that its nearest mode belongs to another cluster rather than its current one, reallocate the object to that cluster and update the modes of both clusters.

4. Repeat 3 until no object has changed clusters after a full cycle test of the whole data set.

3.5.2 Hierarchical Methods

A hierarchical clustering method works by grouping data objects into a hierarchy or “tree” of clusters. Even though we don’t need to partition our data points (totes) into groups at different levels, the agglomerative process of a hierarchical method is very similar to the heuristic F#1. A hierarchical method can be either agglomerative or divisive, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion. An agglomerative hierarchical clustering method uses a bottom-up strategy. It typically starts by letting each object form its own cluster and iteratively merges clusters into larger and larger clusters, until all the objects are in a single cluster or certain termination conditions are satisfied.

A tree structure called a dendrogram is commonly used to represent the process of hierarchical clustering. It shows how objects are grouped together (in an agglomerative method) or partitioned (in a divisive method) step-by-step. The following Figure 3.3 is an example by applying the agglomerative hierarchical clustering method on one of our 20-totes-data set. The vertical axis shows the similarity scale between clusters.

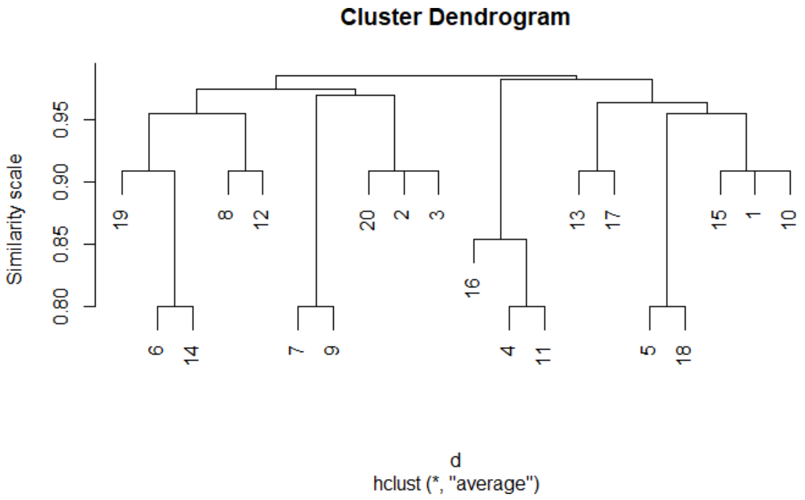


Figure 3.3 Hierarchical cluster dendrogram.

The core need of a hierarchical method is to measure the distance between two clusters, where each cluster is generally a set of objects. Four widely used measures for distance between clusters are as follows, where $d(a_i, a_j)$ is the distance between two objects (totes), a_i and a_j; q_c is the mean for cluster, Qc; and t_c is the number of objects in Qc. They are also known as linkage measures.

Minimum distance: $dist_{min}(Q_{c1}, Q_{c2}) = \min_{a_i \in Q_{c1}, a_j \in Q_{c2}} \{d(a_i, a_j)\}$ (3.14)

Maximum distance: $dist_{max}(Q_{c1}, Q_{c2}) = \max_{a_i \in Q_{c1}, a_j \in Q_{c2}} \{d(a_i, a_j)\}$ (3.15)

Mean distance: $dist_{mean}(Q_{c1}, Q_{c2}) = d(q_{c1}, q_{c2})$ (3.16)

Average distance: $dist_{avg}(Q_{c1}, Q_{c2}) = \frac{1}{t_{c1} t_{c2}} \sum_{a_i \in Q_{c1}, a_j \in Q_{c2}} d(a_i, a_j)$ (3.17)

When an algorithm uses the minimum distance (3.14) to measure the distance between clusters, it is sometimes called a nearest-neighbor clustering algorithm. Moreover, if the clustering process is terminated when the distance between nearest clusters exceeds a user-defined threshold, it is called a single-linkage algorithm. If we view the data points (totes) as nodes of a graph, with edges forming a path between the nodes in a cluster, then the merging of two clusters, Q_{c1} and Q_{c2}, corresponds to adding an edge between the nearest pair of nodes in Q_{c1} and Q_{c2}. Because edges linking clusters always go between distinct clusters, the resulting graph will generate a tree. Thus, an agglomerative hierarchical clustering algorithm that uses the minimum distance measure is also called a minimal spanning tree algorithm.

When an algorithm uses maximum distance (3.15) to measure the distance between clusters, it is sometimes called a farthest-neighbor clustering algorithm. If the clustering process is terminated when the maximum distance between nearest clusters exceeds a user-defined threshold, it is called a complete-linkage algorithm. The distance between two clusters is determined by the most distant nodes (totes) in the two clusters.

The previous minimum and maximum measures represent two extremes in measuring the distance between clusters. They tend to be overly sensitive to outliers or noisy data. The use of mean (3.16) or average distance (3.17) is a compromise between the minimum and maximum distances and overcomes the outlier sensitivity problem. And the average distance is advantageous in that it can handle categorical as well as numeric data. In our case, after trying all these four measures, the average distance gives the best results in clustering our tote-order data sets, so we use this measures in our application of hierarchical method.

3.6 Numerical Study and Benchmark Evaluation

In this section, we evaluate the performance of the fast heuristic F#1 in solving the tote assignment problem, compare the shipping boxes generated by F#1 with the optimal values of MIP and the results of the other two clustering algorithms, k-means and hierarchical methods.

3.6.1 Software and Solver

We use AMPL, a mathematical programming language, to build the integer programming models in SolverStudio (Package Version 0.09.03) and run by CPLEX solver on NEOS (Network-Enabled Optimization System) server. AMPL is an algebraic modeling language to describe and solve high-complexity problems for large-scale mathematical computing (i.e., large-scale optimization and scheduling-type problems). It provides access to more than 60 state-of-the-art solvers in more than a dozen optimization categories and offers a variety of interfaces for accessing the solvers to enable jobs run on distributed high-performance machines. Many modern solvers available on the NEOS Server accept AMPL input. According the NEOS Server is a free internet-based service for solving numerical optimization problems.to the NEOS statistics AMPL is the most popular format for representing mathematical programming problems. Unlike the free student version of AMPL (this version is limited to 500 variables and constraints for linear problems), AMPL on NEOS have no problem size limitations, other than limits on the size of the model and data files that are exchanged with NEOS. SolverStudio is an add-in for Excel that allows you to build and solve optimization models in Excel using many optimization modeling languages. SolverStudio allows models built using AMPL to be solved using the NEOS server. We choose “AMPL on NEOS” as modelling language in SolverStudio, then when we click Solve, SolverStudio will take the model and the associated data from the present spreadsheet and send these off to the NEOS server at neos-server.org. The NEOS server will then run the files. SolverStudio waits for NEOS to report that the run has finished, then takes the model results and writes them back into the spreadsheet.

The F#1 is coded in RStudio (Version 1.1.423). RStudio is a free and open-source integrated development environment (IDE) for R, a programming language and software environment for statistical computing and graphics. We also implement k-means and hierarchical clustering algorithm in RStudio.

3.6.2 Simulation Data Sets

We randomly generate testing data sets based on the following parameter configurations:

6 sizes of tote-order data set, with correlated number of consolidators.

Table 3.3 Problem Sizes of Data Sets

Consolidators	*Totes*	*Orders*
5	**20**	**60**
6	**30**	**80**
7	**35**	**100**
10	**50**	**140**
15	**75**	**210**
20	**100**	**280**

4 cases are considered for number of items in each order correlated with

$$\sum_{i=1}^{n} a_{i,r} = 2 \ or \ 3$$

Table 3.4 Order Cases of Data Sets

Consolidators	*Totes*	*Orders*	*3-item Order %*			
			case #1	*case #2*	*case #3*	*case #4*
5	**20**	**60**	0%	33%	67%	100%
6	**30**	**80**	0%	25%	50%	100%
7	**35**	**100**	0%	25%	50%	100%
10	**50**	**140**	0%	25%	50%	100%
15	**75**	**210**	0%	33%	67%	100%
20	**100**	**280**	0%	25%	50%	100%

3 levels of totes correlated to each other in tote-order matrix are organized for totes.

Easy: every two (or three) totes have same orders to make the whole matrix highly correlated.

Medium: half of the matrix is highly correlated as easy level, the orders in the other half of the matrix is randomly distributed.

Hard: orders in the whole matrix randomly distributed.

3.6.3 Results and Evaluation

Using the data sets described above, we numerically compared the performance of the proposed fast heuristic F#1 with that of the CPLEX solver on NEOS server, moreover, with that of the k-means and hierarchical clustering methods.

By default, jobs submitted to the NEOS Server are assigned as long jobs, which means that they can run at most 8 hours. Jobs submitted to the NEOS Server are also limited to 3 GB of Random-access memory (RAM). It is often MIP problems have issues with memory. MIP solvers accumulate and store in memory information about the branch-and-cut tree as they progress. At some point, the amount of memory required for the tree information may exceed the amount of memory available. Jobs will be terminated due to exceeding the allowable memory limit. We summarize computational time of all test cases running as long jobs by CPLEX solver on NEOS in the Table 3.5 below. For many test cases, CPLEX failed to find the optimal solution within the allowable memory limit on the NEOS server (NA: exceed the allowable memory limit). And for the cases that CPLEX are able to find the optimal solution, the maximum running time reaches to 2.5 hours. In contrast, all instances can be solved by heuristic F#1 within 1 minute of CPU time. In fast fulfillment operations, we are often required to offer instant response, and we may not have the time to wait for CPLEX to provide us with an optimal solution.

Table 3.5 CPLEX Solver Running Time (Miniutes)

Consolidators	*Totes*	*Orders*	*case #1*			*case #2*			*case #3*			*case #4*		
			e	*m*	*h*	*e*	*m*	*h*	*e*	*m*	*h*	*e*	*m*	*h*
5	**20**	**60**	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	3	< 1	< 1	7
6	**30**	**80**	< 1	<1	4	< 1	< 1	21	< 1	< 1	NA	<1	19	NA
7	**35**	**100**	< 1	2	59	< 1	4	NA	< 1	7	NA	<1	NA	NA
10	**50**	**140**	< 1	145	NA	3	NA	NA	7	NA	NA	NA	NA	NA
15	**75**	**210**	< 1	NA	NA	NA	NA	NA	NA	NA	NA	< 1	NA	NA
20	**100**	**280**	< 1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Jobs submitted to the NEOS Server can be assigned as short jobs, which means that they can run at most 5 minutes. In our experiments, we run all 72 cases as both long jobs and short jobs. For the cases that CPLEX are able to find the optimal solution within the allowable memory limit, the solution quality provided by a long-job run has no significant difference to a short-job run. For the cases that CPLEX failed to find the optimal solution within the allowable memory limit, we used the best feasible CPLEX solutions obtained within 5 minutes as a surrogate for the optimal solution. Table 3.6 shows the results obtained using the CPLEX solver on NEOS server.

Table 3.6 CPLEX Solver Results

Consolidators	*Totes*	*Orders*	*case #1*			*case #2*			*case #3*			*case #4*		
			e	*m*	*h*	*e*	*m*	*h*	*e*	*m*	*h*	*e*	*m*	*h*
5	**20**	**60**	60	78	93	69	91	105	68	105	117	68	109	131
6	**30**	**80**	86	107	122	90	116	130	90	117	142	88	147	169
7	**35**	**100**	107	134	152	111	146	165	126	158	190	134	194	222
10	**50**	**140**	150	192	229	157	213	259	158	209	278	161	266	338
15	**75**	**210**	225	301	354	246	336	411	244	436	469	230	430	541
20	**100**	**280**	300	408	476	316	450	530	350	506	589	379	650	838

For each instance, we calculate the empirical error gap, and it defined as:

$$Empirical\ error\ gap = \frac{\zeta(heuristics) - \zeta(OPT)}{\zeta(OPT)} \tag{3.18}$$

$$heuristics = \{F\#1,\ hierarchical,\ kmeans\} \tag{3.19}$$

Where $\zeta(OPT)$ stands for the optimal results obtained using the CPLEX solver to solve problem P defined by (3.1) - (3.6), and $\zeta(heuristics)$ stands for the results obtained by the fast heuristic *F#1*, k-means and hierarchical clustering methods. The value of error gap could be negative in case CPLEX didn't get the optimal solution within the allowable memory limit. Technically, the smaller the error gap the better, which implies a better solution benchmarked against that of CPLEX.

The numerical study results demonstrate a strong performance of the fast heuristic F#1 we developed in both accuracy and efficiency. We observe an overall average error gap of 4.37% from the optimal or the surrogate of the optimal solution over 72 test cases. Meanwhile, we also observed that the average error gap slightly decreased as the size of problem instances increasing, mainly because CPLEX on NEOS server often failed to find the optimal solutions. Our heuristic shows adequate advantages over the CPLEX when the error gap and computational time are jointly considered.

3.7 Fast Heuristic F#2 and Performance Evaluation

The performance objective of our order consolidation / tote assignment problem is to minimize the number of packages and balance the packaging station workload. Some uneven tote assignment solutions may have a smaller number of packages, but the tote quantity balance will bring the objective results up. Therefore, when we applied the k-means and the hierarchical clustering algorithm on the data sets, these existed well-known clustering algorithms didn't perform better than the F#1 heuristic. In order to further improve the overall performance, we developed the fast heuristic F#2. A numerical study is also conducted to demonstrate the performance of the proposed heuristics.

3.7.1 Fast Heuristic F#2 Development

In light of the results reported in section 3.6, F#1 demonstrate a strong performance in accuracy and efficiency. We followed the same structure to develop the F#2. In each iteration, firstly calculate the Jaccard dissimilarity between totes, then select a pair of totes according to the combined effects of Jaccard dissimilarity and the totes quantity balance, assign this pair of totes into one cluster, merge the two totes into one if the cluster hasn’t reach the limit and continue clustering with other totes, repeat the procedure until all totes grouped into clusters. We use a small tote-order matrix (n totes m orders) as the example and describe F#2 as follows.

Tote-Order Matrix *ai,r*

		Orders															
		O_{1}	O_{2}	O_{3}	O_{4}	O_{5}	O_{6}	O_{7}	O_{8}	O_{9}	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}	O_{16}
Totes	T_{1}	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	0
	T_{2}	1	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0
	T_{3}	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1
	T_{4}	0	1	0	1	0	0	0	1	0	1	0	0	0	0	1	0
	T_{5}	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0
	T_{6}	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1
	T_{7}	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0
	T_{8}	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1

Step 1:

Calculate Jaccard dissimilarity $d_{i,j}$ **(Lemma 1)** for all totes i, j∈{1…n}, i ≠ j in the tote-order matrix (For readability, we do not show the $d_{j,i}$ entries; the matrix is symmetric.);

Jaccard Dissimilarity *di,j*

		Tote *j*							
		T_{1}	T_{2}	T_{3}	T_{4}	T_{5}	T_{6}	T_{7}	T_{8}
Tote *i*	T_{1}								
	T_{2}	0.89							
	T_{3}	0.75	1.00						
	T_{4}	0.89	0.89	0.89					
	T_{5}	0.89	0.57	1.00	0.57				
	T_{6}	0.89	1.00	0.75	1.00	1.00			
	T_{7}	0.75	0.89	0.89	0.89	1.00	0.75		
	T_{8}	1.00	0.57	0.89	1.00	0.89	0.75	0.75	

Also calculate the number of totes $t_{i,j}$ if totes i and j are clustered, for all totes i, j∈{1…n}, i ≠ j (For readability, we do not show the $d_{j,i}$ entries; the matrix is symmetric.);

Number of Totes *ti,j* if Tote *i* and *j are Clustered*

		Tote *j*							
		T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
Tote *i*	**T1**								
	T2	2							
	T3	2	2						
	T4	2	2	2					
	T5	2	2	2	2				
	T6	2	2	2	2	2			
	T7	2	2	2	2	2	2		
	T8	2	2	2	2	2	2	2	

Step 2:

In order to minimize the number of packages and the tote quantity balance, clustering totes with minimal dissimilarity and absolute difference from the clustered number of totes to the even number of totes, $min\ \{d_{i,j} + \delta * abs(t_{i,j} - \frac{n}{k})\}$ **(Lemma 2);**

$abs(t_{i,j} - \frac{n}{k})$ indicates the absolute difference between the number of totes within a cluster and the even number of totes for each consolidator; Uneven tote quantity solutions will be effectively decreased by minimizing this absolute difference in each iteration. *n* is the number of totes (*n = 8* in this example) and *k* is the number of consolidators (*k = 3* in this example); *δ* is a constant coefficient, since totes dissimilarity range is [0, 1], we set *δ = 0.1* to bring down the numerical digit of tote quantity balance;

$d_{i,j} + \delta * abs(t_{i,j} - n/k), \delta = 0.1$

		Tote *j*							
		T1	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
Tote *i*	**T1**								
	T2	0.96							
	T3	0.82	1.07						
	T4	0.96	0.96	0.96					
	T5	0.96	0.64	1.07	0.64				
	T6	0.96	1.07	0.82	1.07	1.07			
	T7	0.82	0.96	0.96	0.96	1.07	0.82		
	T8	1.07	0.64	0.96	1.07	0.96	0.82	0.82	

If there's a tie for tote i, j, i', j', $i \neq j \neq i' \neq j'$,

$$\{d_{i,j} + \delta * abs\left(t_{i,j} - \frac{n}{k}\right)\} = \{d_{i',j} + \delta * abs\left(t_{i',j} - \frac{n}{k}\right)\} = \{d_{i',j} + \delta * abs\left(t_{i',j} - \frac{n}{k}\right)\}$$

Select either $\{d_{i\prime,j} + \delta * abs(t_{i',j} - \frac{n}{k})\}$ or $\{d_{i,j'} + \delta * abs\left(t_{i,j'} - \frac{n}{k}\right)\}$ **(Lemma 3)**;

From the above table, we can see if tote *T2* and *T5* are clustered in the earlier iteration, we may lose the advance of both tote *T2*, *T8* clustered together and tote *T4*, *T5* clustered together in the later iterations.

Step 3:

Merge the clustered tote i and j with the $min\ \{d_{i,j} + \delta * abs(t_{i,j} - \frac{n}{k})\}$ (from step 2) into a new tote i', according to the logic:

$$If\ a_{i,r} = a_{j,r} = 0, then\ a_{i\prime,r} = 0$$

$$Else, a_{i\prime,r} = 1$$

For example, if tote T2, T8 clustered together in the previous step, they will be merged into tote T2,8 as follows.

T2	1	0	1	0	0	0	1	1	0	0	0	0	0	1	0	0
T8	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	1

T2,8	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	1

Step 4:

If the number of totes within one cluster hasn't reach the even number of totes for each consolidator n/k, add the merged tote i' to the tote-order matrix and remove the original tote i and j; Else, remove the totes i and j from the tote-order matrix.

Tote-Order Matrix *ai,r*

		Orders															
		O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
Totes	T1	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	0
	T3	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0	1
	T4	0	1	0	1	0	0	0	1	0	1	0	0	0	0	1	0
	T5	0	0	1	1	0	0	0	1	0	0	0	0	0	1	1	0
	T6	0	0	0	0	1	0	0	0	1	0	1	0	1	0	0	1
	T7	1	1	0	0	1	1	0	0	0	0	0	0	1	0	0	0
	T2,8	1	0	1	0	0	0	1	1	0	0	0	0	1	1	0	1

Number of Totes *ti,j* if Tote *i* and *j are Clustered*

		Tote *j*						
		T1	T3	T4	T5	T6	T7	T2,8
Tote *i*	T1							
	T3	2						
	T4	2	2					
	T5	2	2	2				
	T6	2	2	2	2			
	T7	2	2	2	2	2		
	T2,8	3	3	3	3	3	3	

Repeat above steps until all totes are assigned to consolidators.

3.7.2 Lemma Proof

Lemma 1: In each iteration, if $d_{i,j}$ is minimal, then $\sum_{r=1}^{m} z_{c,r}$ is minimal for $x_{i,c} = x_{j,c} = 1$.

Proof:

$$d_{i,j} = \frac{\sum_{r=1}^{m}(a_{i,r} = 1 \;\&\; a_{j,r} = 0) + \sum_{r=1}^{m}(a_{i,r} = 0 \;\&\; a_{j,r} = 1)}{\sum_{r=1}^{m}\left(a_{i,r} = a_{j,r} = 1\right) + \sum_{r=1}^{m}(a_{i,r} = 1 \;\&\; a_{j,r} = 0) + \sum_{r=1}^{m}(a_{i,r} = 0 \;\&\; a_{j,r} = 1)}$$

$$sim(i,j) = 1 - d_{i,j} =$$

$$= \frac{\sum_{r=1}^{m}(a_{i,r} = a_{j,r} = 1)}{\sum_{r=1}^{m}\left(a_{i,r} = a_{j,r} = 1\right) + \sum_{r=1}^{m}(a_{i,r} = 1 \;\&\; a_{j,r} = 0) + \sum_{r=1}^{m}(a_{i,r} = 0 \;\&\; a_{j,r} = 1)}$$

For totes i, j, i',

If $\sum_{r=1}^{m}\left(a_{i,r} = a_{j,r} = 1\right) = \sum_{r=1}^{m}(a_{i,r} = a_{i\prime,r} = 1) = \sum_{r=1}^{m}(a_{i\prime,r} = a_{j,r} = 1)$

And

$\sum_{r=1}^{m}\left(a_{i,r} = a_{j,r} = 1\right) + \sum_{r=1}^{m}(a_{i,r} = 1 \,\&\, a_{j,r} = 0) + \sum_{r=1}^{m}(a_{i,r} = 0 \,\&\, a_{j,r} = 1)$

is minimal, then $sim(i, j)$ is maximal, $d_{i,j}$ is minimal.

For $x_{i,c} = x_{j,c} = 1$,

$$\sum_{r=1}^{m} z_{c,r} = \sum_{r=1}^{m}\left(a_{i,r} = a_{j,r} = 1\right) + \sum_{r=1}^{m}(a_{i,r} = 1 \,\&\, a_{j,r} = 0) + \sum_{r=1}^{m}(a_{i,r} = 0 \,\&\, a_{j,r} = 1)$$

is also minimal.

Lemma 2: In each iteration, if $abs(t_{i,j} - \frac{n}{k})$ is minimal, then b_c is minimal for

$x_{i,c} = x_{j,c} = 1.$

Proof:

In each iteration, $t_{i,j}$ is the number of totes if totes *i* and *j* are clustered, for all totes *i*, *j∈{1…n}, i ≠ j;*

$$For\ x_{i,c} = x_{j,c} = 1, t_{i,j} = \sum_{i=1}^{n} x_{i,c}$$

$$\min\left\{abs\left(t_{i,j} - \frac{n}{k}\right)\right\} = \min\ \{abs\left(\sum_{i=1}^{n} x_{i,c} - \frac{n}{k}\right)\}$$

In the MIP model, we have constraints (3.4) and (3.5) as follows:

$$b_c \geq \sum_{i=1}^{n} x_{i,c} - n/k$$

$$b_c \geq\ n/k - \sum_{i=1}^{n} x_{i,c}$$

Enforce the following equation:

$$b_c = abs(\sum_{i=1}^{n} x_{i,c} - \frac{n}{k})$$

Then

$$\min\{b_c\} = \min\left\{abs\left(t_{i,j} - \frac{n}{k}\right)\right\}$$

Lemma 3: In each iteration, if there’s a tie of minimum for tote $i, j, i', j', and\ i \neq j \neq i' \neq j'$,

$$\{d_{i,j} + \delta * abs\left(t_{i,j} - \frac{n}{k}\right)\} = \{d_{i',j} + \delta * abs\left(t_{i',j} - \frac{n}{k}\right)\} = \{d_{i',j} + \delta * abs\left(t_{i',j} - \frac{n}{k}\right)\}$$

Select either $\{d_{i',j} + \delta * abs(t_{i',j} - \frac{n}{k})\}\ or \left\{d_{i,j'} + \delta * abs\left(t_{i,j'} - \frac{n}{k}\right)\right\}$.

Proof:

Select $\{d_{i,j} + \delta * abs(t_{i,j} - \frac{n}{k})\}\ for\ x_{i,c} = x_{j,c} = 1$, will lead to eliminate opportunities for both $x_{i\prime,c} = x_{j,c} = 1\ and\ x_{i,c} = x_{j\prime,c} = 1$ in the following iterations.

3.7.3 Performance Evaluation

In this section, we regenerated test data sets to evaluate the fast heuristic F#2. For the comparability and sufficiency of the results, we increased the test cases for each size of the original data set.

3.7.3.1 Design of Experiments. The results from previous numerical study show the optimal solutions are difficult to be found when the size of test data set increasing to 10 consolidators, 50 totes, and 140 orders. Therefore, we defined the data set size above this as large data sets and below this as small data sets. In the meanwhile, the results from the Table 3.3 show during the 24 large data sets in the medium and hard level, only one medium level test case is able to get the optimal solution within the allowable memory limit and its running time is more than two hours. Therefore, among the regenerated data sets, we only ran the small data sets by CPLEX solver on NEOS server to get optimal solutions, and test F#1, F#2, hierarchical and k-means algorithm on both small and large data sets. The summary of the data sets generation shown in the following table.

Table 3.7 Summary of Small and Large Data Sets

	Consolidators	*Totes*	*Orders*	*Number of Test Cases*	*Optimal Results Availability*	*3-item Order %*	*Matrix Complexity*	
Small Data Sets	**5**	**20**	**60**	42	YES	Progressive Increasing by 5%	Medium	Hard
	6	**30**	**80**	28				
	7	**35**	**100**	17				
Large Data Sets	**10**	**50**	**140**	22	NO	Progressive Increasing by 10%	Medium	Hard
	15	**75**	**210**	22				
	20	**100**	**280**	22				

3.7.3.2 Coefficient of Tote Quantity Balance. When solving the MIP model for optimal results, we set the balance coefficient β as 1 at the beginning, and increased it as the data set size, 3-item order quantity, and matrix complexity increasing. In this part, we developed an estimation function to calculate the balance coefficient for the large data sets, since the optimal results are not available.

As we mentioned in the section 3.3, the tote quantity balance is to restrict the extremely uneven totes assignment solution that all totes are assigned to one consolidator. Therefore, the upper bound of the tote quantity balance can be calculated by the following function:

$$\frac{n}{k}(k-1)+\left(n-\frac{n}{k}\right)$$

Equivalent to:

$$2n(1-\frac{1}{k})$$

Where n is total number of totes, k is total number of consolidators. The optimal results $\zeta(OPT)$ can be represented by a function related with the total number of orders m and the upper bound of the tote quantity:

$$\zeta(OPT)=m+\alpha\beta 2n(1-\frac{1}{k})$$

Then the coefficient and can be derived as:

$$\alpha = \frac{\zeta(OPT) - m}{\beta 2n(1 - \frac{1}{k})}$$

$$\beta = \frac{\zeta(OPT) - m}{\alpha 2n(1 - \frac{1}{k})}$$

We have two type of orders in the data sets, 2-item orders and 3-item orders. According to the 3-item orders percentage of each data set, we can calculate the total number of items in all of the orders as:

$$3m\pi + 2\mathrm{m}(1 - \pi)$$

The optimal results $\zeta(OPT)$ also related to the total number of items with coefficient ε:

$$\zeta(OPT) = \varepsilon[3m\pi + 2\mathrm{m}(1 - \pi)]$$

Combined with the function we derived above, the estimation function of balance coefficient β is:

$$\beta = \frac{\varepsilon[3m\pi + 2\mathrm{m}(1 - \pi)] - m}{\alpha 2n(1 - \frac{1}{k})}$$

After solved optimal results of all 87 small data sets, we calculated the coefficient α and ε for every case. Then the weighted average value of them, $\alpha = 0.507$ and $\varepsilon = 0.644$, are used to derive the coefficient β for all 66 large data sets.

3.7.3.3 Results. We evaluated performances of the fast heuristic F#2, F#1, hierarchical and k-means algorithm on both small and large data sets.

For the small data sets, we calculated the empirical error gap and performed the one sample t-test. We used the same function as (3.18) to calculate the error gap for results obtained by F#1, F#2, hierarchical and k-means.

$$Empirical\ error\ gap = \frac{\zeta(heuristics) - \zeta(OPT)}{\zeta(OPT)}$$

$$heuristics = \{F\#1,\ F\#2,\ hierarchical,\ kmeans\}$$

From the small data set results Table 3.8, we can see the mean error gap of F#2 is 4.149% and one sample t-test show the true mean is significant below 5%. We also performed the power analysis to validate the power of the t-test. In the meanwhile, mean error gaps of the other methods results are all above 5%.

Table 3.8 Result Summary of 87 Small Data Sets

	Mean of Error Gap	***One Sample t-test***						***t-test Power Calculation***
		Null Hypothesis	***Alternative Hypothesis***	***t***	***df***	***p-value***	***Conclusion***	
F#2	4.149%	True Mean = 5%	True Mean < 5%	-2.798	86	0.00318	Reject the Null Hypothesis in Favor of the Alternative Hypothesis	0.8709144
F#1	8.487%		True Mean > 5%	5.412		< 0.00001		0.9999019
Hierarchical	6.729%		True Mean > 5%	3.709		0.00018		0.9790506
k-means	11.618%		True Mean > 5%	15.49		< 0.00001		1

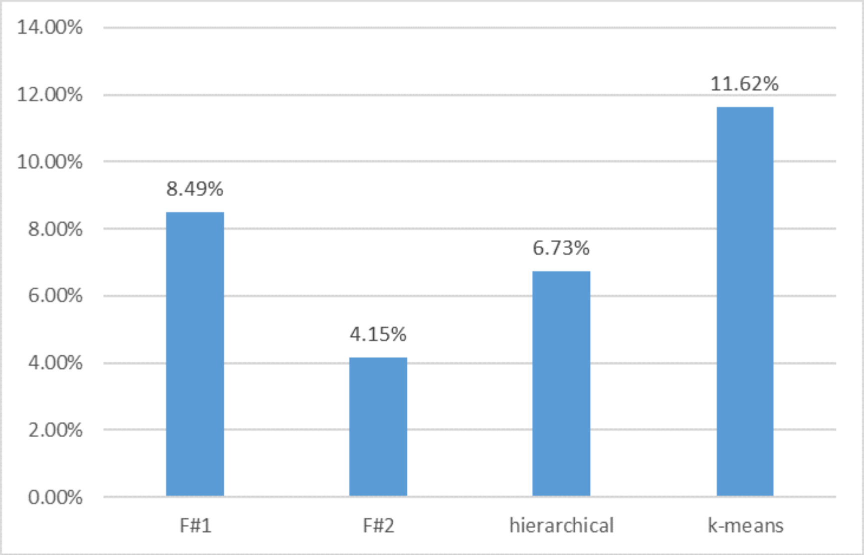


Figure 3.4 Mean error gap of small data sets.

Since the optimal results for the large data sets are not available, we calculated the results differences between F#2 and other methods, which show the results of F#1, hierarchical and k-means are larger than F#2 by 7%, 6% and 10%. We also performed the paired t-test to show the true differences are significant. The result summery Table 3.9 as follows.

$$Results\ Differences\ Percentage\ = \frac{\zeta(Other\ Algorithm) - \zeta(\ F\#2)}{\zeta(\ F\#2)}$$

$$Other\ Algorithm = \{F\#1,\ Hierarchical,\ kmeans\}$$

Table 3.9 Result Summary of 66 Large Data Sets

	Mean of Differences	***Null Hypothesis***	***Alternative Hypothesis***	***Paired t-test***			***Conclusion***	***t-test Power Calculation***
				t	***df***	***p-value***		
F#1 - F#2	26.262 (7%)	True Difference in Means is 0	True Difference in Means is Greater Than 0	12.254	65	< 0.00001	Reject the Null Hypothesis in Favor of the Alternative Hypothesis	1
Hierarchical - F#2	25.157 (6%)			10.388				
k-means - F#2	37.321 (10%)			12.575				

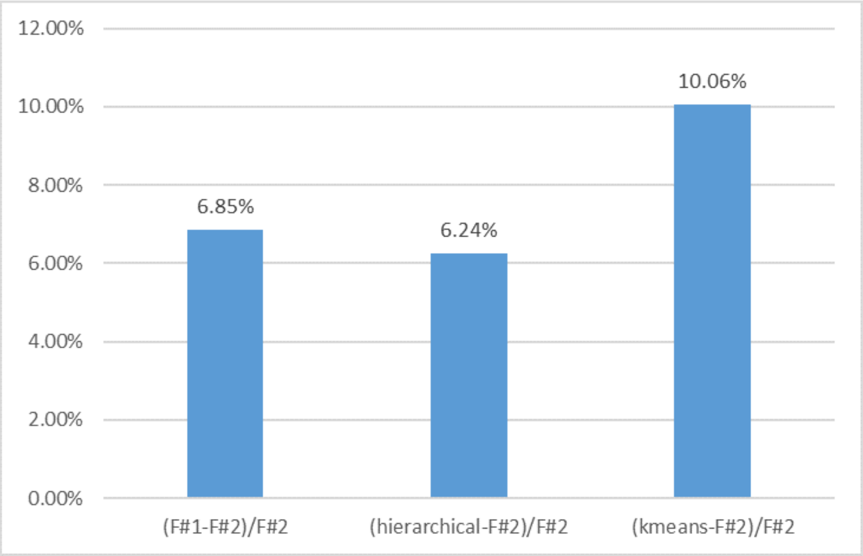


Figure 3.5 Mean algorithms results difference of large data sets.

CHAPTER 4

PERFORMANCE BEHAVIOR ANALYSIS OF FAST HEURISTICS

The order consolidation is a dynamic process, items belong to different orders carrying by different totes come to the consolidation station continuously. In this chapter, performance behavior of the heuristics is further studied as a function of size of the tote-order matrix, multi-item order complexity of the tote-order matrix, number of consolidators and the consolidation batch window.

4.1 Size of Tote-Order Matrix

The size of the tote-order matrix is a critical factor to the complexity of the consolidation problem. Given each three sizes tote-order matrix for both small and large problems in the Chapter 3, we expand three extra-large sizes of tote-order matrix in this chapter, which are shown in the Table 4.1. These three sizes of problems combine with the following design of multi-item order complexity, we generate a variety of data sets and perform experimental tests on them.

Table 4.1 Extra-Large Problem Size

Number of Totes (n)	*Number of Orders (m)*	*Number of Consolidators (k)*
200	560	40
300	840	60
400	1120	80

4.2 Multi-Item Order Complexity of Tote-Order Matrix

A tote-order matrix is a binary matrix with rows as totes and columns as orders. A binary value in a tote-order matrix $a_{i,r} = 1$ means SKU item of order *r* picked in tote *i*. The summation of each column $\sum_{i=1}^{n} a_{i,r}$ stands for SKU items of order *r* are in how many totes. To simplify, we call orders have items in more than one totes as multi-item order. For example, an order has items in three totes, $\sum_{i=1}^{n} a_{i,r} = 3$, this order is called 3-item order. The 3-item shows the order complexity in the tote-order matrix, even though this order may have more than three items.

We include three kinds of multi-item orders, 2, 3 and 4-item orders, in experimental deign of tote-order matrix. The quantity of each kind of multi-item orders take a certain proportion out of the total number of orders (m). We design a set of quantity percentage combination and list in the Table 4.2. The purpose of this design of experiments is to gradually increase complexity of the tote-order matrix by increasing quantity of 4-item orders.

Table 4.2 Multi-Item Orders Quantity Percentage

	Multi-Item Orders		
	2-item	***3-item***	***4-item***
Case #0	100%	0%	0%
Case #1	65%	30%	5%
Case #2	55%	30%	15%
Case #3	45%	30%	25%
Case #4	35%	30%	35%
Case #5	25%	30%	45%
Case #6	15%	30%	55%
Case #7	5%	30%	65%

For each size of tote-order matrix, there are 8 cases of multi-item orders quantity percentage combination. And for each case, we generate 10 random test cases. There are total 240 test cases in 24 conditions. The Table 4.3 shows the total number of items (I) and number of items per order (I/m).

We run the fast heuristic F#2, F#1, hierarchical and k-means algorithm on all test cases. The following Table 4.4, 4.5, and 4.6 show the results of three problem sizes by eight order complexity cases, each value is mean of 10 test cases results. The fast heuristic F#2 performs better than the other three algorithms in all test cases. And the optimal solutions for these extra-large problems are not available. We calculate the results differences percentage to show the difference between the fast heuristic F#2 and all other algorithms.

$$Results\ Differences\ Percentage\ = \frac{\zeta(Other\ Algorithm) - \zeta(\ F\#2)}{\zeta(\ F\#2)}$$

$$Other\ Algorithm = \{F\#1,\ Hierarchical,\ kmeans\}$$

Table 4.3 Total Number of Items (I)

	Tote-Order Matrix Size			***Items per Order (I/m)***
	200 × 560	***300 × 840***	***400 × 1120***	
Case #0	1120	1680	2240	2
Case #1	1344	2016	2688	2.4
Case #2	1456	2184	2912	2.6
Case #3	1568	2352	3136	2.8
Case #4	1680	2520	3360	3
Case #5	1792	2688	3584	3.2
Case #6	1904	2856	3808	3.4
Case #7	2016	3024	4032	3.6

Table 4.4 Results of 200×560 Tote-Order Matrix by 40 Consolidators

	F#1	***F#2***	***Hierarchical***	***kmeans***	***(F#1-F#2)/F#2***	***(Hierarchical-F#2)/F#2***	***(kmeans-F#2)/F#2***
Case #0	1002	958	975	1022	4.67%	1.81%	6.74%
Case #1	1169	1146	1176	1187	1.95%	2.60%	3.56%
Case #2	1253	1231	1277	1311	1.74%	3.69%	6.51%
Case #3	1342	1320	1369	1364	1.67%	3.73%	3.35%
Case #4	1433	1412	1459	1485	1.47%	3.34%	5.20%
Case #5	1530	1506	1547	1542	1.57%	2.69%	2.39%
Case #6	1621	1603	1647	1664	1.15%	2.76%	3.80%
Case #7	1711	1698	1741	1777	0.74%	2.53%	4.66%

Table 4.5 Results of 300×840 Tote-Order Matrix by 60 Consolidators

	F#1	***F#2***	***Hierarchical***	***kmeans***	***(F#1-F#2)/F#2***	***(Hierarchical-F#2)/F#2***	***(kmeans-F#2)/F#2***
Case #0	1521	1443	1477	1542	5.40%	2.38%	6.87%
Case #1	1791	1740	1791	1816	2.92%	2.91%	4.36%
Case #2	1914	1870	1938	1996	2.38%	3.69%	6.76%
Case #3	2046	2011	2079	2085	1.76%	3.40%	3.71%
Case #4	2187	2146	2225	2266	1.91%	3.68%	5.59%
Case #5	2329	2296	2362	2360	1.47%	2.91%	2.78%
Case #6	2477	2436	2510	2544	1.65%	3.02%	4.40%
Case #7	2616	2586	2661	2722	1.16%	2.88%	5.24%

Table 4.6 Results of 400×1120 Tote-Order Matrix by 80 Consolidators

	F#1	F#2	Hierarchical	kmeans	(F#1-F#2)/F#2	(Hierarchical-F#2)/F#2	(kmeans-F#2)/F#2
Case #0	2046	1928	1979	2076	6.07%	2.62%	7.65%
Case #1	2418	2336	2401	2444	3.52%	2.80%	4.66%
Case #2	2583	2520	2606	2685	2.47%	3.40%	6.53%
Case #3	2767	2710	2809	2814	2.11%	3.65%	3.83%
Case #4	2949	2892	3007	3054	1.97%	3.98%	5.60%
Case #5	3145	3088	3195	3184	1.85%	3.47%	3.11%
Case #6	3341	3285	3386	3434	1.72%	3.08%	4.54%
Case #7	3529	3477	3588	3668	1.48%	3.19%	5.50%

The Figure 4.1 illustrate the mean algorithms results difference of all extra-large data sets. Comparing to the large data sets results in the Chapter 3, the extra-large size tote-order matrix with increased order complexity make the differences between the fast heuristic F#2 and all other algorithms reduced. The performance of the k-means still has the largest difference from the fast heuristic F#2. The general performances of the two heuristics #1 and #2 are closer, especially when the order complexity increased from the case #0 to case #7.

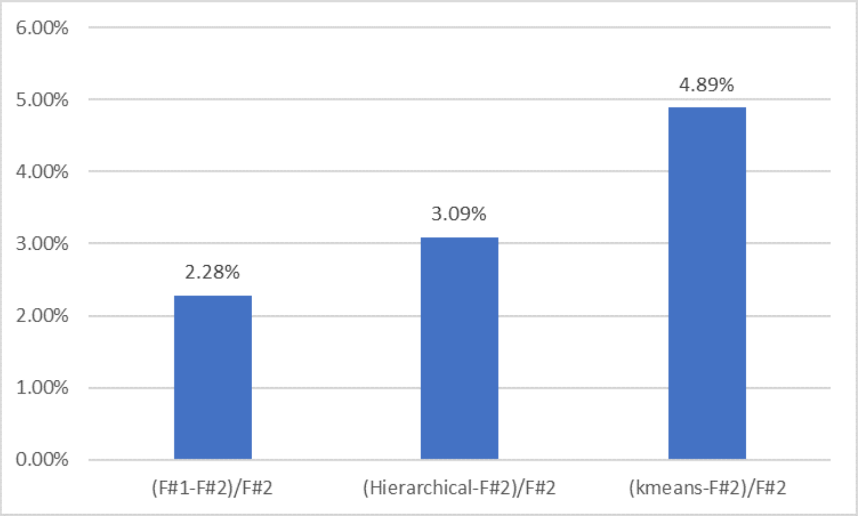


Figure 4.1 Mean algorithms results difference of all extra-large data sets.

4.3 Number of Consolidators

With the 240 tote-order matrices in the extra-large data sets, performance behavior of heuristic F#2 is further studied by changing number of consolidators (k). As shown in the Table 4.7, we run the heuristic F#2 on each size of the tote-order matrix with a set of number of consolidators.

Table 4.7 Number of Consolidators Set for each size of Tote-Order Matrix

Number of Totes (n)	*Number of Orders (m)*	*Number of Consolidators (k)*							
200	**560**	40	35	30	25	20	15	10	5
300	**840**	60	50	45	38	30	22	15	7
400	**1120**	80	70	60	50	40	30	20	10

Table 4.8 shows results run by F#2 on all 200×560 tote-order matrices with different number of consolidators. In this table, each row has 10 random generated tote-order matrices with the same order complexity, each value is the mean of the 10 results. The difference among values of each row shows the effect of different number of consolidators. From forty consolidators to five consolidators, more totes will be assigned to each consolidator, the number of delivery packages will be closer to the optimal solution which is the number of orders (m).

For each tote-order matrix, total number of items (I) is the maximum or worst result that a heuristic solution could get, while number of orders (m) is the minimum or best result. We develop a Maxmin ratio as surrogate for the heuristic F#2 solution performance.

$$Maxmin\,ratio \;=\frac{I-\zeta(\,F\#2)}{I-m}$$

Table 4.9 shows the Maxmin ratio of F#2 results on 200 totes by 560 orders data sets. They also are illustrated by Figure 4.2.

Table 4.8 Mean Number of Delivery Packages run by *F#2* on 200×560 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	40	**35**	**30**	**25**	**20**	**15**	**10**	**5**
Case #0	958	947	939	933	918	899	868	799
Case #1	1146	1130	1113	1105	1082	1049	1013	927
Case #2	1231	1213	1193	1183	1161	1126	1082	983
Case #3	1320	1301	1283	1269	1239	1204	1153	1043
Case #4	1412	1392	1370	1353	1328	1285	1232	1098
Case #5	1506	1481	1463	1444	1415	1362	1304	1148
Case #6	1603	1580	1557	1536	1500	1447	1378	1214
Case #7	1698	1670	1645	1623	1587	1526	1452	1271

Table 4.9 Maxmin Ratio of 200×560 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	40	**35**	**30**	**25**	**20**	**15**	**10**	**5**
Case #0	29%	31%	32%	33%	36%	39%	45%	57%
Case #1	25%	27%	29%	31%	33%	38%	42%	53%
Case #2	25%	27%	29%	30%	33%	37%	42%	53%
Case #3	25%	27%	28%	30%	33%	36%	41%	52%
Case #4	24%	26%	28%	29%	31%	35%	40%	52%
Case #5	23%	25%	27%	28%	31%	35%	40%	52%
Case #6	22%	24%	26%	27%	30%	34%	39%	51%
Case #7	22%	24%	25%	27%	29%	34%	39%	51%

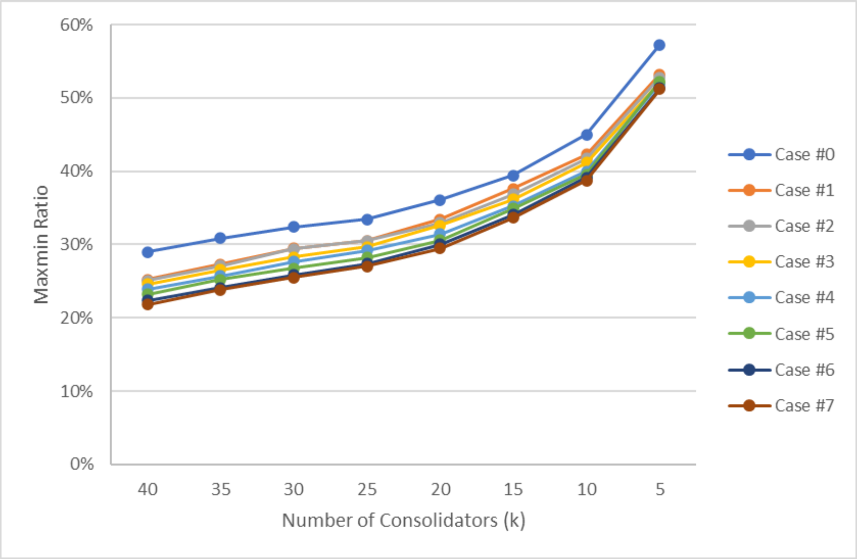


Figure 4.2 Maxmin ratio of 200 totes by 560 orders matrix with different number of consolidators.

The detailed results of 300 totes by 840 orders data sets are shown in the Tables 4.10 and 4.11, illustrated by Figure 4.3.

Table 4.10 Mean Number of Delivery Packages run by *F#2* on 300×840 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	60	**50**	**45**	**38**	**30**	**22**	**15**	**7**
Case #0	1443	1430	1416	1408	1393	1368	1334	1239
Case #1	1740	1718	1697	1683	1653	1612	1565	1452
Case #2	1870	1846	1825	1806	1779	1728	1682	1546
Case #3	2011	1982	1959	1944	1910	1858	1795	1643
Case #4	2146	2116	2091	2073	2040	1987	1916	1737
Case #5	2296	2262	2236	2213	2175	2117	2038	1842
Case #6	2436	2402	2375	2352	2307	2240	2154	1936
Case #7	2586	2546	2518	2490	2444	2370	2281	2037

Table 4.11 Maxmin Ratio of 300×840 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	60	**50**	**45**	**38**	**30**	**22**	**15**	**7**
Case #0	28%	30%	31%	32%	34%	37%	41%	53%
Case #1	23%	25%	27%	28%	31%	34%	38%	48%
Case #2	23%	25%	27%	28%	30%	34%	37%	47%
Case #3	23%	24%	26%	27%	29%	33%	37%	47%
Case #4	22%	24%	26%	27%	29%	32%	36%	47%
Case #5	21%	23%	24%	26%	28%	31%	35%	46%
Case #6	21%	23%	24%	25%	27%	31%	35%	46%
Case #7	20%	22%	23%	24%	27%	30%	34%	45%

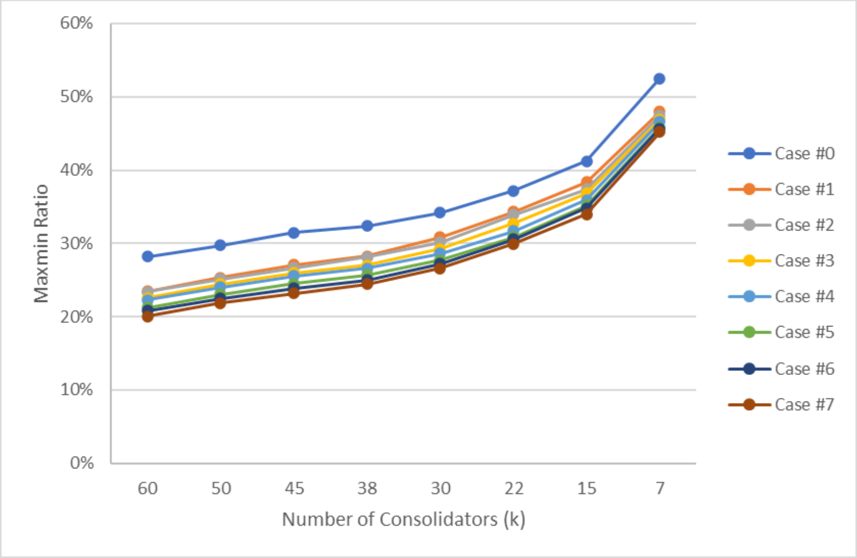


Figure 4.3 Maxmin ratio of 300 totes by 840 orders matrix with different number of consolidators.

The detailed results of 400 totes by 1120 orders data sets are shown in the Tables 4.12 and 4.13, illustrated by Figure 4.4.

Table 4.12 Mean Number of Delivery Packages run by *F#2* on 400×1120 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	80	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Case #0	1928	1912	1901	1887	1870	1838	1805	1708
Case #1	2336	2308	2286	2267	2231	2181	2124	2008
Case #2	2520	2491	2465	2442	2397	2344	2289	2145
Case #3	2710	2676	2648	2619	2581	2520	2451	2298
Case #4	2892	2853	2825	2801	2757	2693	2619	2430
Case #5	3088	3048	3018	2991	2942	2873	2782	2580
Case #6	3285	3240	3206	3181	3128	3045	2956	2730
Case #7	3477	3432	3395	3367	3316	3227	3127	2871

Table 4.13 Maxmin Ratio of 400×1120 Tote-Order Matrix with Different Number of Consolidators

	Number of Consolidators (k)							
	80	**70**	**60**	**50**	**40**	**30**	**20**	**10**
Case #0	28%	29%	30%	32%	33%	36%	39%	48%
Case #1	22%	24%	26%	27%	29%	32%	36%	43%
Case #2	22%	24%	25%	26%	29%	32%	35%	43%
Case #3	21%	23%	24%	26%	28%	31%	34%	42%
Case #4	21%	23%	24%	25%	27%	30%	33%	42%
Case #5	20%	22%	23%	24%	26%	29%	33%	41%
Case #6	19%	21%	22%	23%	25%	28%	32%	40%
Case #7	19%	21%	22%	23%	25%	28%	31%	40%

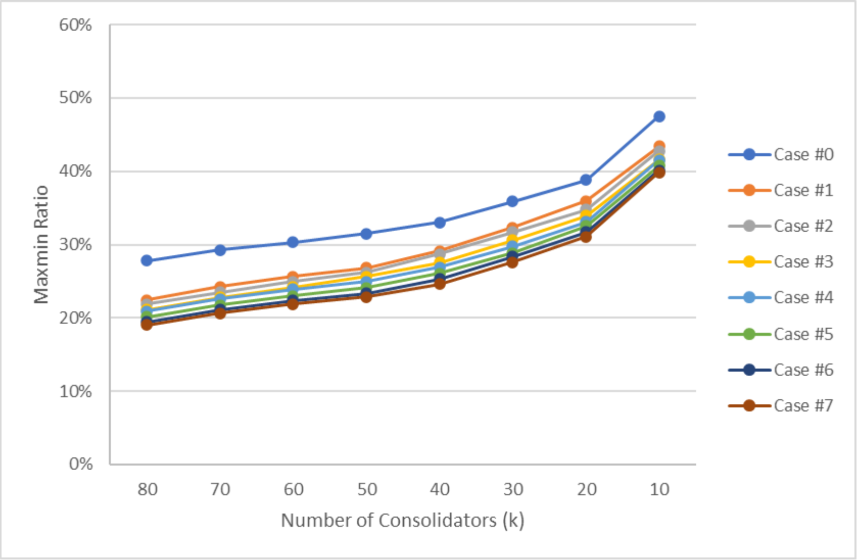


Figure 4.4 Maxmin ratio of 400 totes by 1120 orders matrix with different number of consolidators.

4.4 Consolidation Tote Batch Window

An F-Warehouse continuously generates picklist totes. At any time, a batch of totes are to be processed through several available order packaging stations. In the above sections of this chapter, we generate extra-large sizes of tote-order matrix and perform tote consolidation to the whole matrix. When considering the waiting time for the consolidation batch window, larger batch size leads longer waiting time. The size of the tote batch window is a partitioning decision on the dynamic flow of totes. In this section, we generate extra-large sizes of tote-order matrix to simulate the tote flow, then divide the matrix into smaller tote batches to apply the heuristic F#2 and analyze the results.

4.4.1 Dynamic Design of Tote-Order Matrix

In the previous tote-order matrix generation, there is no rule for items of same order could be in which tote, they totally random distribute between the first tote and the last tote of the matrix. In this section, we design the dynamic tote flow which has two key features:

The range between two items of one order cannot exceed 50 totes.

The relationship of tote i out of n totes and order r out of m orders is:

$i/n = r/m$

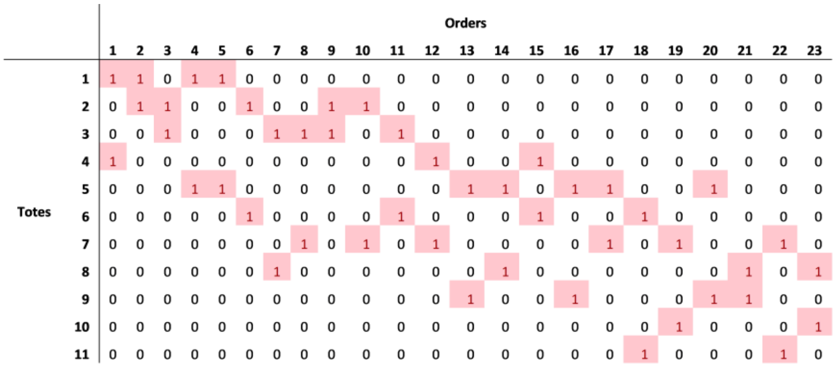
We perform the following detailed steps to generate the extra-large tote-order matrix, each order r has two different items located in tote i and j:

Step 1: assign the first item of each order r to tote i, which is a random tote from the range between $\frac{r}{m} * (n - 50) - 10$ and $\frac{r}{m} * (n - 50) + 10$; make sure the number of items in totes $\sum_{r=1}^{m} a_{i,r}$ not exceed the $2m/n$.

Step 2: assign the first item of each order r to tote j, the distance between tote i and j cannot exceed 50. If $a_{i,r} = a_{j,r} = 1$, $|i - j| \leq 50$, for all $r;$ make sure the number of items in totes $\sum_{r=1}^{m} a_{i,r}$ not exceed the $2m/n$.

Step 3: perform data cleaning, first merge totes only having one or two items with other totes, then remove empty totes.

The Figure 4.5 shows a small portion of a tote-order matrix with the dynamic tote flow design. For visualization purpose, the tote range between two items of one order is set as 5 totes instead of 50 in this example.



		Orders																						
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
Totes	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	0	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
	5	0	0	0	1	1	0	0	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	0
	6	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0
	7	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1	0	1	0	0	1	0
	8	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1
	9	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0
	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0

Figure 4.5 Example of tote-order matrix with dynamic design.

We generate extra-large tote-order matrix in three sizes, 560, 840 and 1120 orders, 10 random matrices for each size. These 10 random matrices have different number of totes due to the dynamic design and data cleaning step. The Table 4.14

below shows sizes of all 30 tote-order matrices.

Table 4.14 Size of 30 Tote-Order Matrices

Number of Orders (m)	Number of Totes (n) for Each 10 Random Matrices										Mean Number of Totes (n)
560	174	172	174	178	171	177	174	174	175	172	174
840	266	263	267	266	262	264	269	270	266	264	266
1120	360	362	363	361	356	360	361	358	360	363	360

4.4.2 Experimental Design and Result Analysis

In the previous sections, we increase the number of consolidators according to the size of the tote-order matrix to test the heuristics performance of handling the large problem size. In this part of design, we treat the extra-large data sets as the continuous tote flow and fix the number of consolidation stations as 5. The decision variable is the consolidation batch window size, how many totes as a batch that be distributed to the 5 available consolidation stations.

For each extra-large size of tote-order matrix, we set the smallest batch window as 30 totes and gradually increase 30 totes for each batch size until the batch size cover the whole tote-order matrix. We apply the fast heuristic F2 to calculate the number of delivery packages of each batch and add up all batches for the whole tote-order matrix. Mean results of the 10 random tote-order matrices of each size for all batch sizes are shown in the Table 4.15. As batch size increase, the number of delivery packages decrease, which is expected since larger batch size means more totes assigned to consolidators and more orders consolidated. But large batch size leads more waiting time for each tote batch, and the consolidation station capability also limit the tote batch size.

To further demonstrate the results trending and the solution performance, we calculate the Maxmin ratio and show in the Table 4.16. Since each order has two items, the total number of items I is two times of the number of orders m. The Figure 4.6 illustrate the Maxmin ratio results in all variations of batch size.

Table 4.15 Mean Number of Delivery Packages run by F#2 with Different Batch Sizes

Number of Orders (m)	Mean Number of Totes (n)	Batch Size (Number of Totes)											
		30	60	90	120	150	180	210	240	270	300	330	360
560	174	1062	945	878	889	877	788						
840	266	1557	1432	1322	1324	1252	1245	1225	1211	1125			
1120	360	2046	1861	1763	1708	1698	1619	1610	1602	1583	1555	1530	1445

Table 4.16 Maxmin Ratio of *F#2* Results with Different Batch Sizes

Number of Orders (m)	Mean Number of Totes (n)	Batch Size (Number of Totes)											
		30	60	90	120	150	180	210	240	270	300	330	360
560	174	10%	31%	43%	41%	43%	59%						
840	266	15%	29%	43%	42%	51%	52%	54%	56%	66%			
1120	360	17%	34%	43%	47%	48%	55%	56%	57%	59%	61%	63%	71%

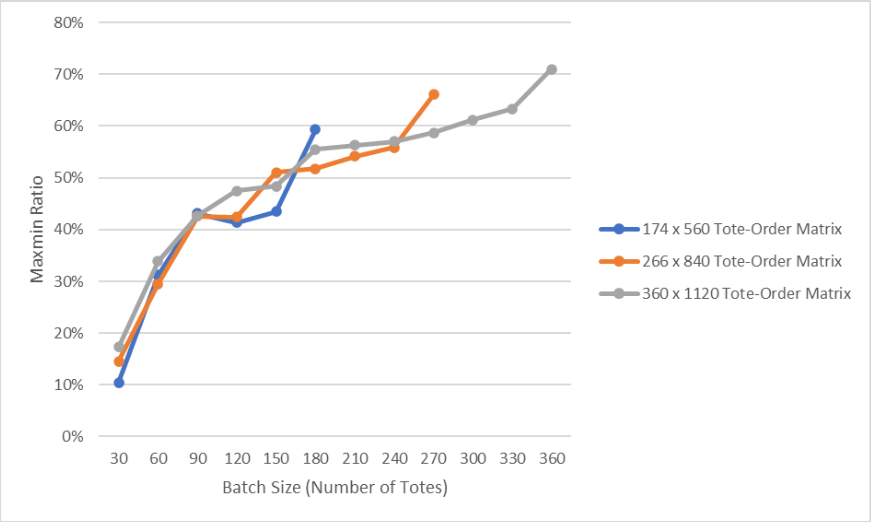


Figure 4.6 Maxmin ratio of F#2 results with different batch sizes.

4.5 Twinning Design – Future Research

The twinning design is the order similarity percentage in the tote-order matrix. In this section, we design two orders could have items in two same totes, which called twinning orders. This is a preliminary design for the future research. The experimental design only includes the data sets of 200×560 Tote-Order Matrix. The percentage of twinning orders quantity out of total 560 orders is from 0%, 20%, 40%... to 100%. For each percentage of twinning orders quantity, 10 random tote-order matrices are generated. The items of each order are in two totes. The total number of items (I) is 1120. We use the same experimental design for number of consolidators as section 4.3. The heuristic #2 is applied on the data sets of tote-order matrix with the variety of number of consolidators.

For each twinning order percentage, mean number of delivery packages of the 10 random tote-order matrices for all consolidator numbers are shown in the Table 4.17. The following table 4.18 show the Maxmin ratio of the F#2 results. And the Figure 4.7 illustrates the Maxmin ratio results for all twinning orders percentages.

Table 4.17 Mean Number of Delivery Packages run by F#2 with Different Number of Consolidators

Twinning Orders Percentage	***Number of Consolidators (k)***							
	40	**35**	**30**	**25**	**20**	**15**	**10**	**5**
100%	821	808	804	799	793	777	761	723
80%	835	821	813	807	799	784	767	727
60%	833	822	813	809	796	782	764	729
40%	870	858	851	845	835	814	793	752
20%	907	900	905	900	901	874	845	776
0%	958	947	939	933	918	899	868	799

Table 4.18 Maxmin Ratio of *F#2* Results with Different Number of Consolidators

Twinning Orders Percentage	***Number of Consolidators (k)***							
	40	**35**	**30**	**25**	**20**	**15**	**10**	**5**
100%	53%	56%	56%	57%	58%	61%	64%	71%
80%	51%	53%	55%	56%	57%	60%	63%	70%
60%	51%	53%	55%	56%	58%	60%	64%	70%
40%	45%	47%	48%	49%	51%	55%	58%	66%
20%	38%	39%	38%	39%	39%	44%	49%	61%
0%	29%	31%	32%	33%	36%	39%	45%	57%

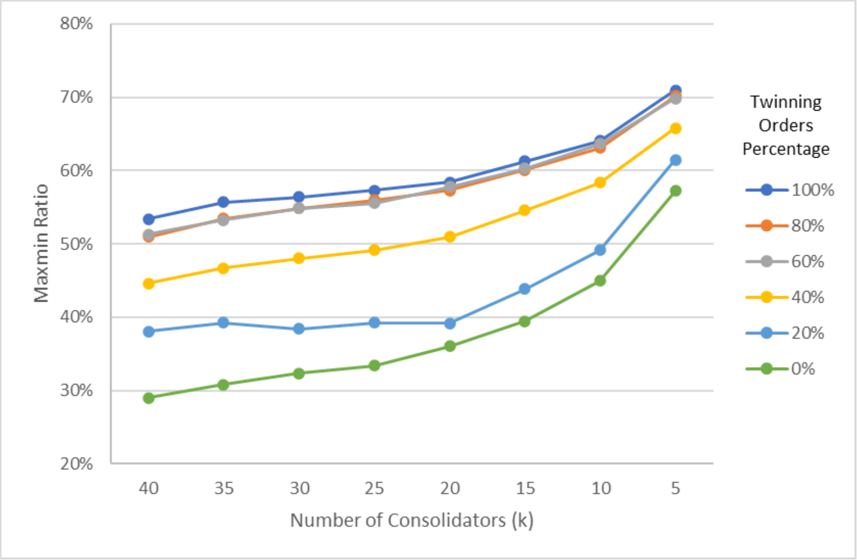


Figure 4.7 Maxmin ratio of F2 results with different number of consolidators.

CHAPTER 5

BUY ONLINE PICKUP FROM STORE (BOPS)

5.1 Introduction

5.1.1 Omnichannel Retailing

The success of Amazon confirms that a singular channel strategy, pure online, can effectively meet the demand requirements of most customers. To expand into the online channel, retailers are increasingly pursing a dual distribution strategy: product inventory is positioned both in stores shelves and in a fulfillment center.

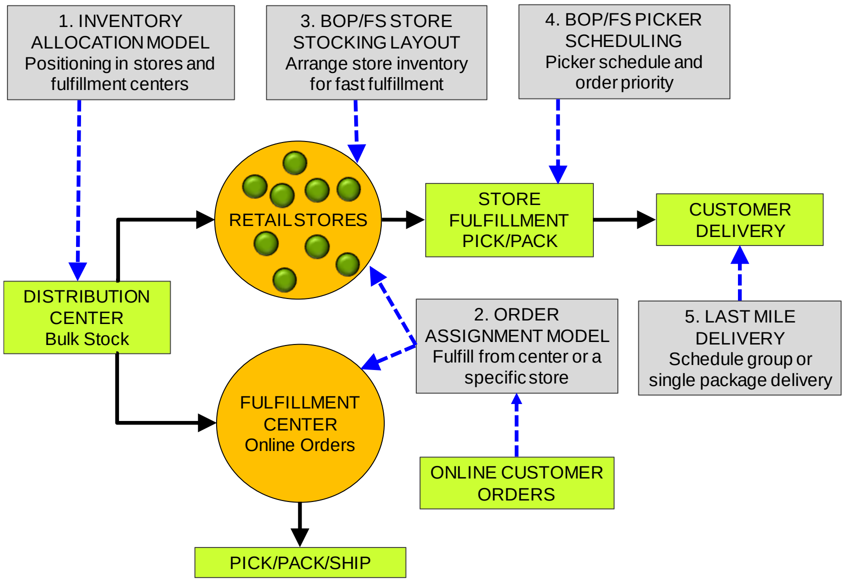

1. INVENTORY ALLOCATION MODEL
Positioning in stores and fulfillment centers
3. BOP/FS STORE STOCKING LAYOUT
Arrange store inventory for fast fulfillment
4. BOP/FS PICKER SCHEDULING
Picker schedule and order priority
RETAIL STORES
STORE FULFILLMENT PICK/PACK
CUSTOMER DELIVERY
DISTRIBUTION CENTER
Bulk Stock
2. ORDER ASSIGNMENT MODEL
Fulfill from center or a specific store
5. LAST MILE DELIVERY
Schedule group or single package delivery
FULFILLMENT CENTER
Online Orders
ONLINE CUSTOMER ORDERS
PICK/PACK/SHIP

Figure 5.1 Operational structure of a dual distribution strategy.

As the Figure 5.1 shows, the omnichannel distribution supply chain consists of a central warehouse that stores large quantities of bulk inventory, several retail stores each of which holds inventory for immediate customer sales, and a fulfillment center which is designed for quick shipment of online orders.

5.1.2 The Store is the Fulfillment Center

An omnichannel retailer with physical stores receives online customer orders through its website. Orders are then directed to a specific company store, where the ordered items are picked from shelf inventory. Picked items are packaged and the package is either (i) Shipped to the customer address – BOFS or (ii) Picked up from the store by the customer – BOPS. S-Strategy fulfillment is a store-based supply chain solution to fulfill online customer orders. Success in online retailing requires a fast fulfillment machine, a system that starts from SUBMIT ORDER and ends with parcel delivery. In an S-Strategy the orders are picked from store inventory and then packaged for customer delivery.

The most aggressive brick-and-mortar retailers, led by Walmart, are using a S-Strategy or BOP/FS model to grow their online business. In effect they have converted the store into a fulfillment center. The key advantage is that there are no transshipments and last mile delivery occurs directly from the store. A second advantage is the ability to offer same day delivery using a local last mile delivery partner. Amazon is also pursuing a similar strategy with the acquisition of Whole Foods.

BOPS and BOFS are the strategy of choice for many retailers, but Sheffi (2016) argue that these solutions are unlikely to provide the needed efficiency gains. He argues that they disrupt store workflow and add inefficient tasks to a site ill-designed for order picking. The efficiency question is: Is an S-Strategy competitive?

This chapter first presents the operational structure of the buy online pickup from store (BOPS) strategy, and second presents two decisions models that are integral to this strategy: BOPS store stocking layout and BOPS picker scheduling.

5.2 Online Order Fulfillment in a BOPS Retailer

Many retailers, particularly in the grocery business, have built their online sales strategy around a BOPS operation. BOPS is an expensive activity, requiring a picker to walk through the store inventory and fulfill a customer's inline order.

5.2.1 BOPS Operational Elements

Forward Stock – Items stocked on shelves in the retail section of the store. Stock is arranged as per the stores product display strategy and area is always populated with browsing customers. The forward stock cannot be physically rearranged, nor can the stocking assignment changed to promote more efficient picking.

Back Stock – Fast pick area in the receiving or rear part of the store. Area is not open to consumer retail. The shelving space is capacitated and only a limited number of SKUs are selectively located here.

Picker Pool – Store employees assigned to online order picking. This is the primary direct cost of BOPS.

Order Queue – Received online orders, time stamped and waiting for picking. Orders will consist of one or more items.

Online Order Pick and Pack – Area where picked orders are packed and then held for delivery.

5.2.2 Online Order Fulfillment Problems

Figure 5.2 illustrate the order picking process in a BOPS retailer. As online orders continuously entering the system, an arrival time is stamped on each order. An order can consist of one or more items. When the picker schedule starts, a picker goes to collect items of one order from the online order pick pack area and come back after picking them all. Then a fulfilled time is stamped to this order. The difference between these two timestamps is the order fulfillment time, and it decided by the pick travel distance and picker schedule.

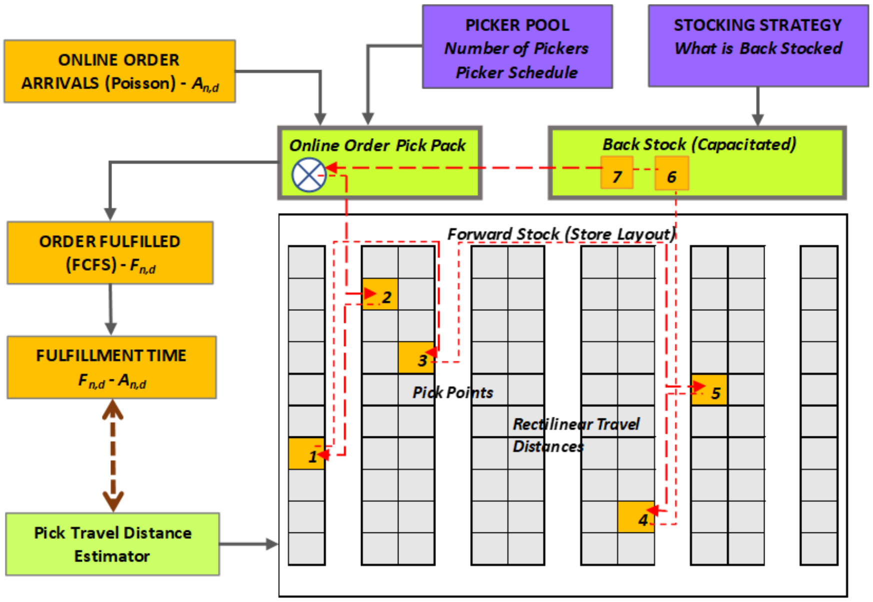


Figure 5.2 Physical configuration of the fulfillment problem.

Store Stocking Layout Problem – The pick travel distance is determined by the stocking layout. The forward stock is the common retail section of a physical store, this section is arranged for customers browsing displayed products and cannot be physically rearranged. The back stock is the fast pick area but the shelving space capacitated, only a limited number of SKUs are selectively located here. The problem is which SKU items should be back stocked. We explicit the item location model and back-stock decision strategy for this problem.

Picker Scheduling Problem – The picker labor cost is the primary direct cost of BOPS. Long picker schedule will increase the labor cost, while short picker schedule will increase the order waiting cost. The picker start time also decides the order fulfilled time. In the picker scheduling problem, we develop the picker schedule optimization model to minimize the BOFS fulfillment cost.

The BOPS goals are, first, minimize the order picking time and, second, minimize the order fulfillment cost. We model the order arrival process and the store inventory dispersion to describe the fulfillment process. Simulation of online order picking is used to compare several decision methods.

Model Notation:

$k \in M$	Items stocked in the store $\{1 \ldots m\}$
(i,j)	Forward stocking location address of item $k \in M$, where i is the aisle-direction number and j is the rack-direction number
$d \in D$	Days in the simulation cycle
$n \in N_d$	Order arriving during the active order period on day d
$A_{n,d}$	Order arrival time
$Q_{n,d}$	Number of different items in an order
$O\{\}_{n,d}$	Set of different items included in order n
$p \in P$	Available order pickers

5.3 BOPS Store Stocking Layout

5.3.1 Forward Stock

We simulated 100 SKU items $k \in \{1 \ldots 100\}$ as a 10-by-10 shelves layout. As the Figure 5.3 shown, the online order pick pack area is located at the back corner of the store. The picker always starts from this area to pick items and comes back to drop off items when picking finished.

Based on this starting point, each item k has a stocking location address (i,j), where i is the aisle-direction number and j is the rack-direction number. The items stocked on racks sharing a same aisle have the same aisle-direction number i. Stocking location address of all 100 SKU items in the Figure 5.4.

From the single item aspect, we use the stocking location $(L_k, k \in M)$ to represent the pick travel distance from the start point to item k:

$$L_k = 2(i_k + j_k)$$

SKU Item - k

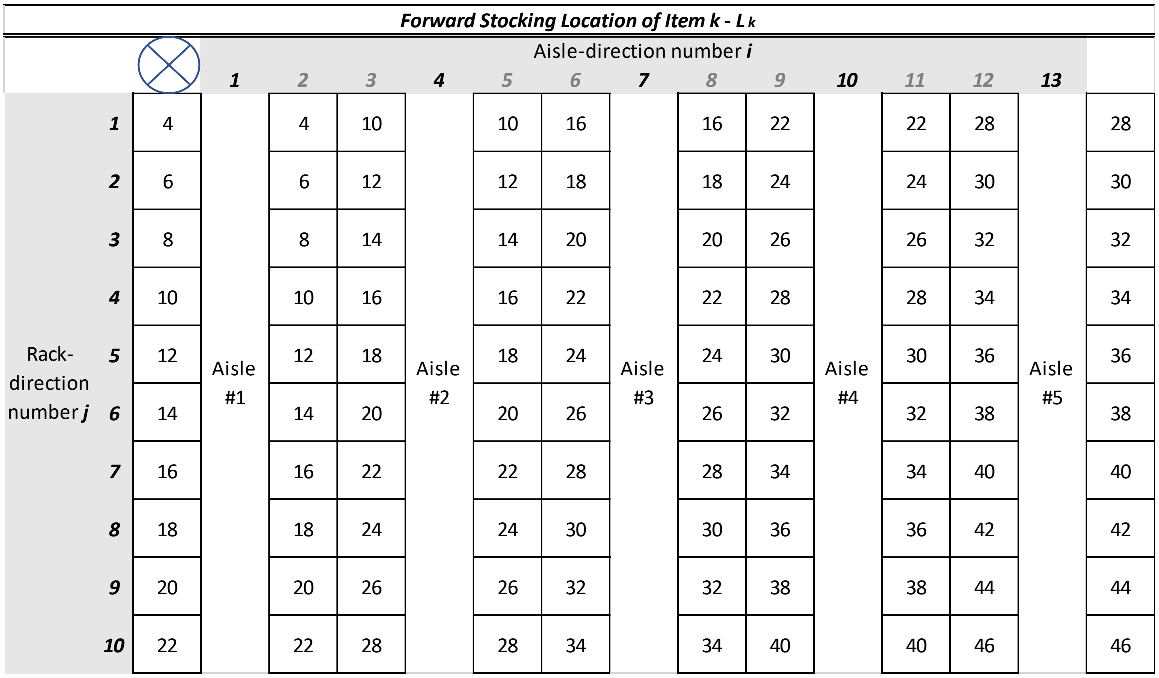
		⊗	Aisle-direction number *i*													
			1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	
	1	1	Aisle #1	11	21	Aisle #2	31	41	Aisle #3	51	61	Aisle #4	71	81	Aisle #5	91
	2	2		12	22		32	42		52	62		72	82		92
	3	3		13	23		33	43		53	63		73	83		93
	4	4		14	24		34	44		54	64		74	84		94
Rack-direction number ***j***	***5***	5		15	25		35	45		55	65		75	85		95
	6	6		16	26		36	46		56	66		76	86		96
	7	7		17	27		37	47		57	67		77	87		97
	8	8		18	28		38	48		58	68		78	88		98
	9	9		19	29		39	49		59	69		79	89		99
	10	10		20	30		40	50		60	70		80	90		100

Figure 5.3 Forward stock of SKU item.

Stocking Location Address - (i,j)

		⊗	Aisle-direction number *i*													
			1	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	
	1	(1,1)	Aisle #1	(1,1)	(4,1)	Aisle #2	(4,1)	(7,1)	Aisle #3	(7,1)	(10,1)	Aisle #4	(10,1)	(13,1)	Aisle #5	(13,1)
	2	(1,2)		(1,2)	(4,2)		(4,2)	(7,2)		(7,2)	(10,2)		(10,2)	(13,2)		(13,2)
	3	(1,3)		(1,3)	(4,3)		(4,3)	(7,3)		(7,3)	(10,3)		(10,3)	(13,3)		(13,3)
	4	(1,4)		(1,4)	(4,4)		(4,4)	(7,4)		(7,4)	(10,4)		(10,4)	(13,4)		(13,4)
Rack-direction number ***j***	***5***	(1,5)		(1,5)	(4,5)		(4,5)	(7,5)		(7,5)	(10,5)		(10,5)	(13,5)		(13,5)
	6	(1,6)		(1,6)	(4,6		(4,6	(7,6)		(7,6)	(10,6)		(10,6)	(13,6)		(13,6)
	7	(1,7)		(1,7)	(4,7)		(4,7)	(7,7)		(7,7)	(10,7)		(10,7)	(13,7)		(13,7)
	8	(1,8)		(1,8)	(4,8)		(4,8)	(7,8)		(7,8)	(10,8)		(10,8)	(13,8)		(13,8)
	9	(1,9)		(1,9)	(4,9)		(4,9)	(7,9)		(7,9)	(10,9)		(10,9)	(13,9)		(13,9)
	10	(1,10)		(1,10)	(4,10)		(4,10)	(7,10)		(7,10)	(10,10)		(10,10)	(13,10)		(13,10)

Figure 5.4 Stocking location address of SKU item.



Forward Stocking Location of Item k - L_k

Aisle-direction number ***i***

		⊗	**1**	2	3	**4**	5	6	**7**	8	9	**10**	11	12	**13**	
	1	4		4	10		10	16		16	22		22	28		28
	2	6		6	12		12	18		18	24		24	30		30
	3	8		8	14		14	20		20	26		26	32		32
	4	10		10	16		16	22		22	28		28	34		34
Rack-direction number ***j***	**5**	12	Aisle #1	12	18	Aisle #2	18	24	Aisle #3	24	30	Aisle #4	30	36	Aisle #5	36
	6	14		14	20		20	26		26	32		32	38		38
	7	16		16	22		22	28		28	34		34	40		40
	8	18		18	24		24	30		30	36		36	42		42
	9	20		20	26		26	32		32	38		38	44		44
	10	22		22	28		28	34		34	40		40	46		46

Figure 5.5 Forward stocking location of SKU item.

5.3.2 Online Order Arrivals

The order arrivals follow the Poisson distribution, with the interarrival time (1/ λ) as 5 minutes. The daily active order period is 6 AM to 8 PM. Orders consist of one or more items $(k \in M)$. For each order, we first generate a random integer number followed the uniform distribution on the interval from 1 to 5, which is the number of different SKU items in an order, Qn,d. Then we generate a set of different items included in order n on day d, O{}n,d.

Item Frequency Design – The item frequency design is to create order designs with different item frequency and quantify the performance difference. We evenly distribute the 100 SKU items into 4 groups. The item grouping is shown in the table 5.1 below. To make sure there is no significant difference about the items pick travel distance among these 4 groups, the summations of stocking location $(\sum L_k)$ for items in groups are close.

Table 5.1 Problem Sizes with Number of Variables and Constrains

	SKU Items - k																									*∑ Lk*
Group #1	1	2	3	4	5	6	7	8	9	10	91	92	93	94	95	96	97	98	99	100	41	51	45	50	60	624
Group #2	11	12	13	14	15	16	17	18	19	20	81	82	83	84	85	86	87	88	89	90	42	52	55	49	59	624
Group #3	21	22	23	24	25	26	27	28	29	30	71	72	73	74	75	76	77	78	79	80	43	53	46	48	58	626
Group #4	31	32	33	34	35	36	37	38	39	40	61	62	63	64	65	66	67	68	69	70	44	54	56	47	57	626

For different designs, we change the percentage of group items out of total orders. Table 5.2 shows the 5 designs we created. In design #1, the 4 groups percentage are same, which means the frequency of all 100 SKU items showing up in total orders are evenly distributed. In the following designs, we gradually increase the percentage of one group items. In design #5, the group #4 items constitute 70 percent of total orders, which makes them have the most frequency.

Table 5.2 Item Frequency Design

	Group #1	***Group #2***	***Group #3***	***Group #4***
Design #1	25%	25%	25%	25%
Design #2	20%	25%	25%	30%
Design #3	15%	20%	25%	40%
Design #4	10%	15%	20%	55%
Design #5	5%	10%	15%	70%

For each design, we generate 20 replications of 1-day order arrivals. There are total 100 order arrival data sets for the stocking layout problem.

5.3.3 Arrange Store Inventory for Fast Fulfillment

The BOPS stocking problem is to select SKU items and stock a part inventory of them in the fast pick area for quick pick, we also call this back-stock strategy. The fast pick area locates in the back part of the store and close to the order pick pack area. When orders have items that back stocked in this area, the picker could get the item immediately instead of traveling around the store, so the item picking time could be saved. The objective of the back-stock strategy is to minimize the order picking time.

Decision Variable:

k∈*B* Items in the back-stock area for quick pick, capacitated and limited by the allocated back stock square footage, and the item inventory.

5.3.3.1 Back Stock Strategy. The space of fast pick area is capacitated and only a limited number of SKU items are selectively located here. In the simulation model, we set 10 SKU items *(k∈B)* can be back stocked out of the total 100 SKU items *(k∈M)*. As the Figure 5.6 shows, we test the following back-stock strategies by running simulation experiments on 1-day orders, the data sets with 5 item frequency designs by 20 replications.

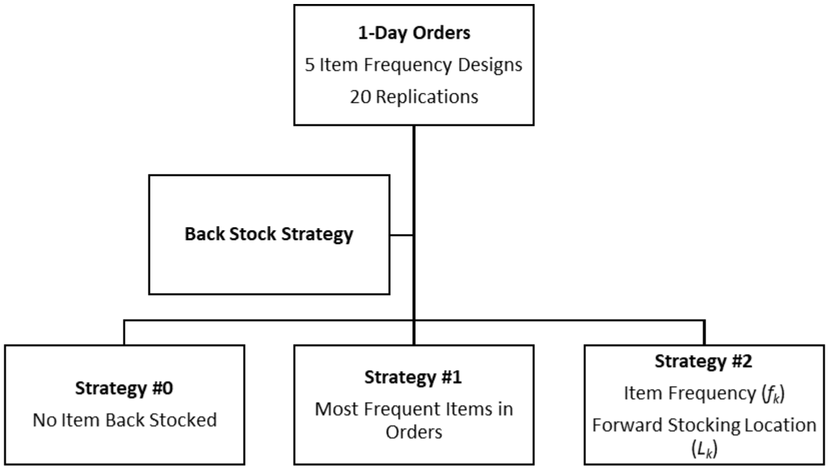


Figure 5.6 Back stock strategy.

Strategy #0 – no item back stocked. This is the nominal case, online orders are just forwarded to the store, B is null.

Strategy #1 – most frequent items in orders. The straightforward back-stock strategy is to select the most frequent items based on the order history. From the order arrivals data, the frequency f_k of all 100 SKU items showing up in orders are available, assign the 10 items with highest frequency to the fast pick area.

Strategy #2 – item frequency and forward stocking location. For each item, the forward stocking location L_k represents the pick travel distance between the item and the order pack area. Considering items with a same frequency f_k from order history, the item with longer pick travel distance has higher priority to be selected into the fast pick area. For all 100 SKU items, in order to normalize these two sets of value, f_k and L_k, to the same scale [0, 1], divide them by the maximum value of each:

$$\frac{f_k}{\max_{k \in M} f_k} + \frac{L_k}{\max_{k \in M} L_k}$$

After the calculation for all items $k \in M$, 10 of them with the largest value are the back stocked items $k \in B$.

5.3.3.2 Pick Travel Distance / Picking Time Estimation. As orders arrival during the active order period, one picker is sending to collect them one by one. Each order is a picklist for one time. The picker starts from the pick pack area, pick all items for the order, come back, and drop the items off for packing.

Model Assumptions:

There is only one picker (P=1).

Orders are processed in the arrival sequence (FCFS).

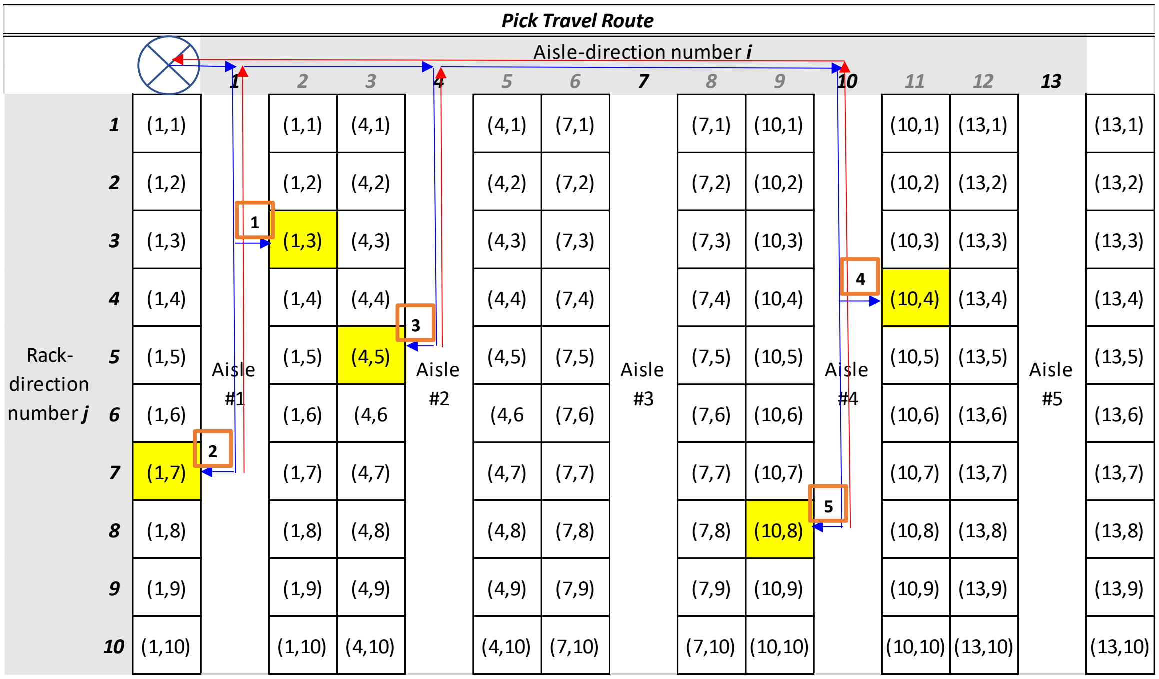
All items in an order are picked in the same picklist.

Items in the back-stock area ($k \in B$) are skipped for pick travel distance estimation.

For a single-item order, the pick travel distance is the stocking location of the item:

$$R_{n,d} = L_k \big| k \in O\{\}_{n,d}, k \notin B$$

The pick travel route of a multi-item order is as the figure 5.7 shown. For example, an order *n,d* has 5 items, $Q_{n,d} = 5$, the stocking location address, $(i_k, j_k) \big| k \in O\{\}_{n,d}$, are highlighted as the below shown. The picker traverses and picks the items from the small number to large number in both aisle and rack direction. The items having a same stocking location address in aisle direction *i* will be picked when the picker go down the aisle. After finishing an aisle, the picker needs to go back to the zero-level of the rack direction j and continues the following aisles.



			1	2	3	4	5	6	7	8	9	10	11	12	13	
	1	(1,1)		(1,1)	(4,1)		(4,1)	(7,1)		(7,1)	(10,1)		(10,1)	(13,1)		(13,1)
	2	(1,2)		(1,2)	(4,2)		(4,2)	(7,2)		(7,2)	(10,2)		(10,2)	(13,2)		(13,2)
	3	(1,3)	1	(1,3)	(4,3)		(4,3)	(7,3)		(7,3)	(10,3)		(10,3)	(13,3)		(13,3)
	4	(1,4)		(1,4)	(4,4)	3	(4,4)	(7,4)		(7,4)	(10,4)	4	(10,4)	(13,4)		(13,4)
Rack-direction number *j*	5	(1,5)	Aisle	(1,5)	(4,5)	Aisle	(4,5)	(7,5)	Aisle	(7,5)	(10,5)	Aisle	(10,5)	(13,5)	Aisle	(13,5)
	6	(1,6)	#1	(1,6)	(4,6	#2	(4,6	(7,6)	#3	(7,6)	(10,6)	#4	(10,6)	(13,6)	#5	(13,6)
	7	(1,7)	2	(1,7)	(4,7)		(4,7)	(7,7)		(7,7)	(10,7)		(10,7)	(13,7)		(13,7)
	8	(1,8)		(1,8)	(4,8)		(4,8)	(7,8)		(7,8)	(10,8)	5	(10,8)	(13,8)		(13,8)
	9	(1,9)		(1,9)	(4,9)		(4,9)	(7,9)		(7,9)	(10,9)		(10,9)	(13,9)		(13,9)
	10	(1,10)		(1,10)	(4,10)		(4,10)	(7,10)		(7,10)	(10,10)		(10,10)	(13,10)		(13,10)

Figure 5.7 Pick travel route of a multi-item order.

Pick travel distance for order n,d:

$$R_{n,d} = 2\left[\max_{k\in O\{\}_{n,d}, k\notin B} i_k + \sum_{k\in O\{\}_{n,d}, k\notin B} \max(j_k|i_k\ with\ same\ value)\right]$$

The order picking time is time duration of traveling and picking items in an order. Then for order n,d, it can be calculated by the pick travel distance $R_{n,d}$ and the number of different items in the order $Q_{n,d}$. With the following parameters:

α – Picker walking time of each unit

β – Picking and packing time of each item

Picking time for order n,d:

$$T_{n,d} = \alpha R_{n,d} + \beta Q_{n,d}$$

5.3.3.3 Performance Analysis of Back Stock Strategy. The objective of the back-stock strategy is to minimize the order picking time. For each order, the picking time is a rectilinear function of the pick travel distance. Since the simulation data sets are 1-day order arrivals, we minimize the summation of the whole day orders pick travel distance.

$$\min \sum_{n \in N_d} R_{n,d}$$

The strategy #0 is the nominal case with no item in the fast pick area, for each replication, we use it as the base line to measure the performance of strategy #1 and #2. The following equations of δ_1 and δ_2 calculate how much the strategy #1 and #2 perform better than the nominal case. They calculate the difference of the total pick travel distance for all orders in day *d* between the strategy #1 (#2) and the strategy #0.

$$\delta_1 = \frac{\sum_{n \in N_d} R_{n,d} \mid Strategy\ \#0 - \sum_{n \in N_d} R_{n,d} \mid Strategy\ \#1}{\sum_{n \in N_d} R_{n,d} \mid Strategy\ \#0}$$

$$\delta_2 = \frac{\sum_{n \in N_d} R_{n,d} \mid Strategy\ \#0 - \sum_{n \in N_d} R_{n,d} \mid Strategy\ \#2}{\sum_{n \in N_d} R_{n,d} \mid Strategy\ \#0}$$

We generate 1-day order arrivals with 5 item frequency designs, and 20 replications for each design. We show the detailed simulation results of design #1 in the Table 5.3 as an example.

In average of the 20 replications, the whole day orders pick travel distance can be decreased 12.42% from the nominal case (strategy #0) by strategy #1, and 17.6% by strategy #2. We are also interested in the performance differences between strategy #1 and #2 ($\delta_2 - \delta_1$). For design #1 orders, mean of the differences between strategy #1 and #2 ($\delta_2 - \delta_1$) is 5.18%. The following paired two sample t-Test supports the differences is statistically significant [t=-10.45, p=2.58E-09]. The strategy #2 performs better than the strategy #1 in design #1 orders.

Table 5.3 Back Stock Strategy Results of Design #1

	$\sum_{n \in N_d} R_{n,d}$					
Replication	**Strategy #0**	**Strategy #1**	**Strategy #2**	***δ_1***	***δ_2***	***δ_2 - δ_1***
1	7850	6586	6312	16.10%	19.59%	3.49%
2	7324	6384	6078	12.83%	17.01%	4.18%
3	8024	6882	6820	14.23%	15.00%	0.77%
4	7808	7020	6612	10.09%	15.32%	5.23%
5	7626	6736	6228	11.67%	18.33%	6.66%
6	8870	7822	7108	11.82%	19.86%	8.05%
7	8814	7512	7290	14.77%	17.29%	2.52%
8	7956	6862	6632	13.75%	16.64%	2.89%
9	8392	7544	6868	10.10%	18.16%	8.06%
10	7464	6320	6032	15.33%	19.19%	3.86%
11	7696	6802	6184	11.62%	19.65%	8.03%
12	7604	6596	6330	13.26%	16.75%	3.50%
13	8030	7142	6678	11.06%	16.84%	5.78%
14	8258	7580	6792	8.21%	17.75%	9.54%
15	8688	7750	7188	10.80%	17.27%	6.47%
16	8294	7300	6818	11.98%	17.80%	5.81%
17	7840	6806	6416	13.19%	18.16%	4.97%
18	8796	7646	7098	13.07%	19.30%	6.23%
19	8226	7080	6796	13.93%	17.38%	3.45%
20	7420	6642	6330	10.49%	14.69%	4.20%

Table 5.4 Paired Two Sample t-Test for Means

	δ1	***δ2***
Mean	0.124151	0.175997
Variance	0.000396	0.000227
Observations	20	20
Pearson Correlation	0.218399	
Hypothesized Mean Difference	0	
df	19	
t Stat	-10.4504	
P(T<=t) one-tail	1.29E-09	
t Critical one-tail	1.729133	
P(T<=t) two-tail	2.58E-09	
t Critical two-tail	2.093024	

We summarize the results of all 5 designs in the Table 5.5. $\overline{\delta_1}$ and $\overline{\delta_2}$ are the mean δ_1 and δ_2 of 20 replications. Means of the differences $(\overline{\delta_2 - \delta_1})$ are statistically significant for all 5 designs. The strategy #2 performs better than the strategy #1 in all 5 designs of orders.

Table 5.5 Summary of Back Stock Strategy Results for 5 Designs

	$\overline{\delta_1}$	$\overline{\delta_2}$	$\overline{\delta_2 - \delta_1}$	*t Stat*	*P(T<=t) two-tail*
Design #1	12.42%	17.60%	5.18%	-10.45	2.58E-09
Design #2	12.30%	16.90%	4.60%	-15.23	4.20E-12
Design #3	13.73%	17.80%	4.07%	-12.09	2.30E-10
Design #4	18.06%	22.40%	4.34%	-11.20	8.24E-10
Design #5	22.87%	28.12%	5.25%	-8.57	5.89E-08

We draw the following box plot to examine experimental results graphically. The Figure 5.8 indicates that both strategy #1 (δ_1) and #2 (δ_2) perform better in design #4 and design #5 than the other designs. The means of design #1, #2, and #3 do not differ.

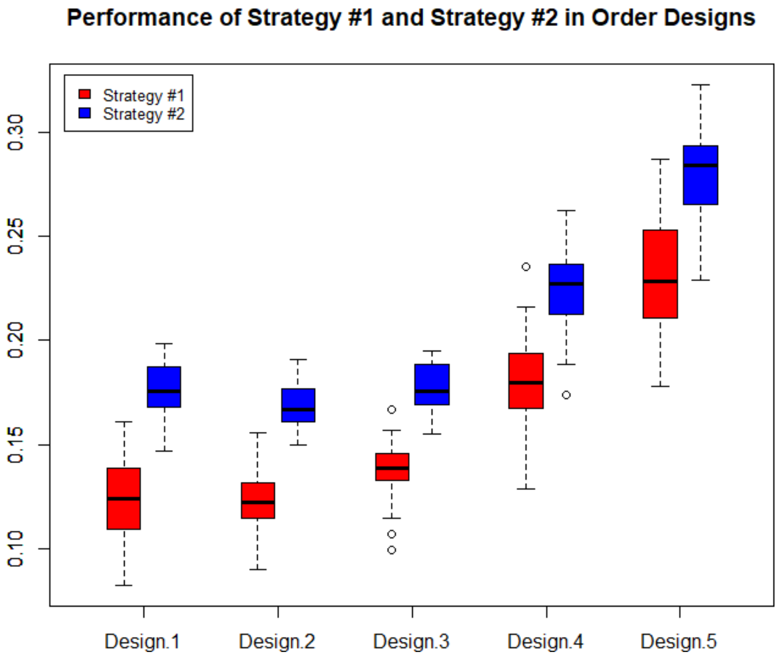


Figure 5.8 Box plot of back stock strategy 1 and 2 performance in 5 designs.

We use the analysis of variance (ANOVA) to test if different order designs affect the mean strategy #2 performance ($\overline{\delta_2}$). As the following Tables 5.6 and 5.7 shown, we have strong evidence to conclude that null hypothesis is not true. That is, the order design affects the strategy #2 performance [F=154.9405, p=9.87E-41].

Table 5.6 Summary of Back Stock Strategy #2

Groups	*Count*	*Sum*	*Average*	*Variance*
Design #1	20	3.519949	0.175997	0.000227
Design #2	20	3.380074	0.169004	0.000136
Design #3	20	3.560724	0.178036	0.000138
Design #4	20	4.480331	0.224017	0.000435
Design #5	20	5.624685	0.281234	0.000522

Table 5.7 ANOVA of Back Stock Strategy #2

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.180701	4	0.045175	154.9405	9.87E-41	2.467494
Within Groups	0.027699	95	0.000292			
Total	0.2084	99				

5.4 BOPS Picker Scheduling

5.4.1 Fulfillment Objective

The primary goal of BOPS picking operation is to minimize the order fulfillment cost. In this section, we define the fulfillment cost objective function with the picker scheduling decision variables and cost coefficients, perform 5-day simulation for the order picking operation, then analyze the simulation results.

Decision Variables:

S_p, E_p Picker schedule – Start time and End Time

$F_{n,d}$ Order fulfill time – Latest pick time of an item in O{}n,d

$X_{n,d}$ Order picked on same day ($X_{n,d}$ =0) or next day ($X_{n,d}$ =1)

Back Stock Setup Cost: An S-Strategy attempts to leverage the existing store stock to fulfill orders, but the layout is inherently inefficient for fast picking. Selected inventory is back stocked for faster picking. This is fixed measure and not in the objective function.

Delivery Tardiness (Waiting) Cost: Difference between the order arrival time and the customer delivery time.

Picker Schedule Cost: The number of order pickers and their work hours.

The delivery tardiness (waiting) cost and the picker schedule cost are variable measures, with the following cost coefficients, they form the objective function of fulfillment cost:

C_h Picker hourly labor cost

C_w Waiting time cost coefficient, hourly delay

C_O Overnight Waiting time cost coefficient

BOPS Fulfillment Cost (daily):

$$\psi = C_h \sum_{p \in P} \left[E_p - S_p\right] + \frac{C_w}{D} \sum_{d \in D} \sum_{n \in N_d P} \left[F_{n,d} - A_{n,d} \,|X_{n,d} = 0\right]$$

$$+ \frac{1}{D} \sum_{d \in D} \sum_{n \in N_d P} \left[C_w\left(20 - A_{n,d}\right) + C_w\left(F_{n,d} - 6\right) + C_O|X_{n,d} = 1\right]$$

The first part of the objective function is the picker schedule cost, it depends on the decision variables picker start time and end time (Sp, Ep). Since the daily order active period is 6 AM to 8 PM, in the simulation experiments all picker schedules $S_p \geq 6$ and $E_p \leq 20$ (24-hour clock). There is only one picker (P=1).

The second part of the objective function is the delivery tardiness (waiting) cost for the same-day fulfilled orders ($X_{n,d}$ = 0). The simulation cycle is 5 days (D=5). Orders are processed in the arrival sequence (FCFS). For order $n \in N_d$, the order fulfill time ($F_{n,d}$) is the time point of the latest pick time of an item in O{}n,d. It can be calculated by the order picking time ($T_{n,d}$), order arrival time ($A_{n,d}$) and the previous order fulfill time ($F_{n-1,d}$).

$$F_{n,d} = A_{n,d} + T_{n,d} \; |A_{n,d} \geq F_{n-1,d}$$

$$F_{n,d} = F_{n-1,d} + T_{n,d} \; |A_{n,d} < F_{n-1,d}$$

The third part of the objective function is the delivery tardiness (waiting) cost for the next-day fulfilled orders ($X_{n,d}$= 1). The orders left to the next day fulfilled have three waiting segments. The same day waiting segment is from the order arrival time ($A_{n,d}$) to the end of the order active period 8 PM (20). The next day waiting segment is from the start of the order active period 6 AM to the order fulfill time ($F_{n,d}$). The middle waiting segment in between is the overnight waiting.

5.4.2 Design of Experiments

The purpose of the experimental design in the picking operation is to investigate the performance sensitivity to picker schedule and waiting cost coefficient. We apply the back-stock strategy #2 since it has the best performance.

5.4.2.1 Order Arrival Process. For the picker scheduling experiments, we generate order arrivals for 5-day planning period. Instead of a fixed arrival rate (λ) for the whole day order arrivals, we create the order arrival surge design. The item frequency design also applied on the 5-day orders. The Figure 5.9 summarizes the order arrivals difference between the stocking and picking problem.

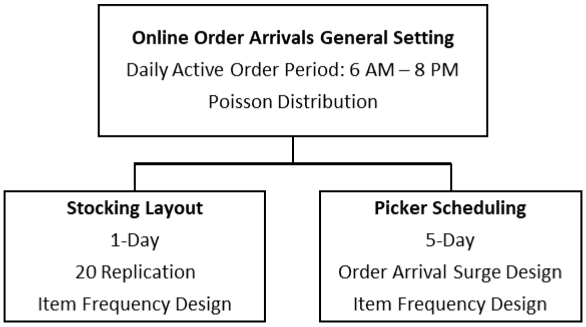


Figure 5.9 Order arrivals difference between the stocking and picking problem.

Order Arrival Surge Design – Instead of setting an arrival rate (λ) for the whole day's active order period, we separate the total 14 hours into 7 time-windows, 2 hours each, to create an order arrival surge during the noon. The Table 5.8 shows the order interarrival time (1/ λ) of every time window in minutes. In the simulation model, time is represented in minutes. For example, 6 AM is represented by 360 and 8 PM is 1200. In the Figure 5.10, we show the frequency of order arrival time in the 5-day simulation to visualize the surge.

Table 5.8 Order Arrival Surge Design

	AM			PM			
Time Window	6-8	8-10	10-12	12-2	2-4	4-6	6-8
Time Window in Minutes	360-480	480-600	600-720	720-840	840-960	960-1080	1080-1200
Order Interarrival Time 1/λ (Minutes)	7	5	4	2	3	4.5	6

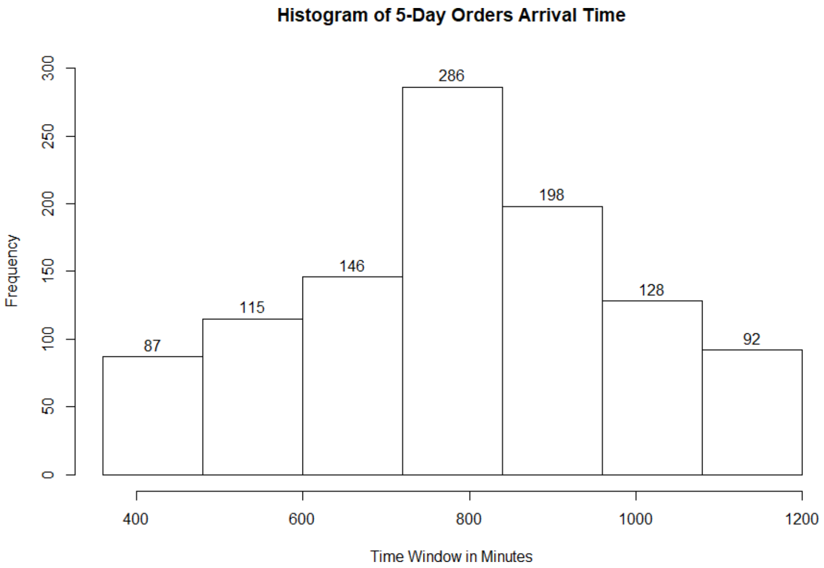


Figure 5.10 Histogram of 5-day orders arrival time.

5.4.2.2 Picker Scheduling. The experimental design of picker schedule involves picker scheduled length or $E_p - S_p$. The longest picker schedule is same as the daily order active period, 14 hours, from 6 AM to 8 PM. We calculate the shortest picker scheduled length by the daily picking time of total orders:

$$\frac{1}{D}\sum_{d \in D}\sum_{n \in N_d P} T_{n,d}$$

In the online order arrivals section, we generate the simulation data of 5-day orders with 5 item frequency designs. After calculation, the average daily picking time of the 5 designs is 7.315 hours. We set the shortest picker scheduled length as 8 hours. The Table 5.9 show the experimental design of picker schedule.

Table 5.9 Picker Schedule

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Picker scheduled Hours ($E_p - S_p$)	8	6 AM - 2 PM	7 AM - 3 PM	8 AM - 4 PM	9 AM - 5 PM	10 AM - 6 PM	11 AM - 7 PM	12 PM - 8 PM
	9	6 AM - 3 PM	7 AM - 4 PM	8 AM - 5 PM	9 AM - 6 PM	10 AM - 7 PM	11 AM - 8 PM	
	10	6 AM - 4 PM	7 AM - 5 PM	8 AM - 6 PM	9 AM - 7 PM	10 AM - 8 PM		
	11	6 AM - 5 PM	7 AM - 6 PM	8 AM - 7 PM	9 AM - 8 PM			
	12	6 AM - 6 PM	7 AM - 7 PM	8 AM - 8 PM				
	13	6 AM - 7 PM	7 AM - 8 PM					
	14	6 AM - 8 PM						

5.4.2.3 Cost Coefficients. The hourly waiting time cost coefficient (C_w) is an indirect or welfare cost, and tricky to estimate. We take the extreme case of 14-hour picker schedule (6 AM – 8 PM) from the above experimental design. This case gives the maximum picker schedule hours and minimum delivery tardiness (waiting) time without any order left to next day. This is the fastest fulfilment, then we think in this case the delivery tardiness (waiting) cost is equivalent to the picker schedule cost. With a given picker hourly labor cost, $C_h = 30$, an estimated waiting time cost ($\widetilde{C_w}$) can be calculated by:

$$\widetilde{C_w} = \frac{14DC_h}{\sum_{d\in D}\sum_{n\in N_{d}P}\left[F_{n,d} - A_{n,d}\right]}$$

We use minute as time unit in the simulation, this estimated waiting time cost ($\widetilde{C_w}$) gives the waiting time cost per minute, then the hourly waiting time cost coefficient (C_w):

$$C_w = 60\widetilde{C_w}\ \forall\ \widetilde{C_w}\epsilon\{0.4, 0.3, 0.2, 0.1, 0.05\}$$

With five sets of the 5-day orders, we run the simulation experiments and calculate the average $\widetilde{C_w}$ is 0.344. Then we expend it to five values and investigate the performance sensitivity to them.

Instead of using full duration from 8 PM to 6 AM as the overnight waiting time, we develop a relationship between overnight waiting time cost coefficient (C_O) and hourly waiting time cost coefficient (C_w) and use γ to stand for the overnight waiting hours. We also test two value of γ.

$$C_O = \gamma C_w \; \forall \; \gamma \epsilon \{2, 1\}$$

5.4.3 Performance Analysis of Simulation Results

5.4.3.1 Picker Scheduling. Based on each item frequency design in orders, we have 5 sets of 5-day order arrivals. We perform the simulation of all experimental picker schedules on each design of 5-day orders. Take the simulation results of design #1 orders as an example, the Table 5.10 summarize the total fulfillment cost of 5-day orders for all picker schedules. The results of the objective function are the summation of picker schedule cost, same-day waiting cost, overnight waiting cost and next-day waiting cost.

The order active period is from 6 AM to 8 PM. When the picker working to the last order arrival (picker end time $E_p = 8\, PM$), almost no order left to the next day, so the overnight and next-day waiting cost is zero. One exception is the 8-hour schedule from 12 PM to 8 PM, with 5 orders left to the next day, the fulfillment cost has a small portion of overnight and next-day waiting cost.

From the Figure 5.11, we can see the fulfillment objective is sensitive to both the picker schedule length and the start time. A later start time is preferred, since it reduce the risk of orders delayed overnight. As the start time continues pushing back, the same-day waiting increases in the queuing model, the fulfillment cost goes up. With $\widetilde{C_w} = 0.4\; and\; \gamma = 2,$ the optimal picker schedule is the 10-hour length from 9 AM to 7 PM.

Table 5.10 BOPS Fulfillment Cost $\left(\widetilde{C_w} = 0.4, \gamma = 2\right)$ – Design #1

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Labor Scheduled Hours ($E_p - S_p$)	**8**	2844	2417	2114	2146	2110	2206	2387
	9	2272	2031	1947	1895	1940	2104	
	10	2033	1945	1859	1815	1893		
	11	1988	1907	1852	1840			
	12	1988	1936	1908				
	13	2048	2017					
	14	2154						

Table 5.11 Same-day Waiting Cost $\left(\widetilde{C_w} = 0.4, \gamma = 2\right)$ – Design #1

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Labor Scheduled Hours ($E_p - S_p$)	**8**	933	637	470	564	658	870	1170
	9	497	333	302	338	484	754	
	10	275	214	191	227	393		
	11	169	127	131	190			
	12	96	81	108				
	13	61	67					
	14	54						

Table 5.12 Overnight Waiting Cost $\left(\widetilde{C_w} = 0.4, \gamma = 2\right)$ – Design #1

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Labor Scheduled Hours ($E_p - S_p$)	**8**	380	271	182	134	78	38	4
	9	261	179	126	76	34	0	
	10	179	126	76	34	0		
	11	126	76	34	0			
	12	76	34	0				
	13	34	0					
	14	0						

Table 5.13 Next-day Waiting Cost $\left(\widetilde{C_w} = 0.4, \gamma = 2\right)$ – Design #1

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Labor Scheduled Hours $(E_p - S_p)$	**8**	331	309	263	247	174	98	12
	9	165	169	168	130	72	0	
	10	80	105	92	54	0		
	11	42	54	37	0			
	12	16	20	0				
	13	3	0					
	14	0						

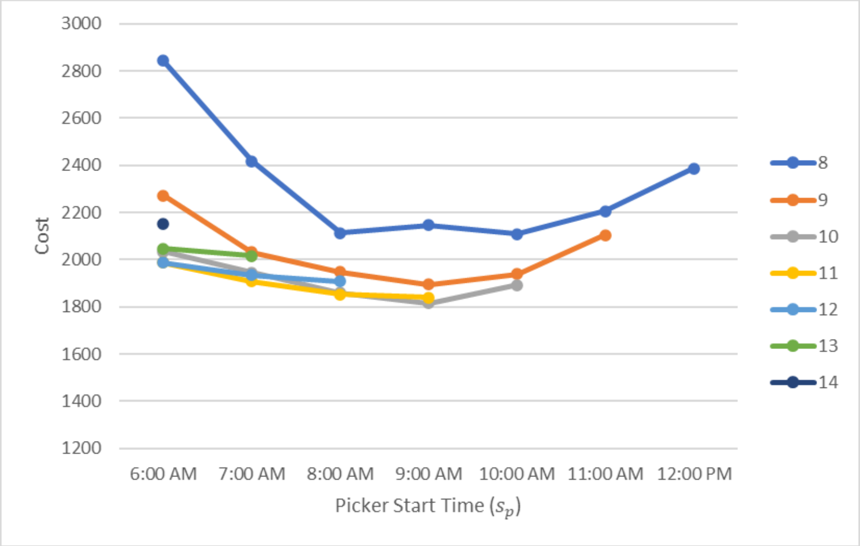


Figure 5.11 BOPS fulfillment cost (waiting time cost coefficient 0.4, overnight waiting 2 hours) – design #1.

5.4.3.2 Order Designs. We show the fulfillment cost of the other 4 designs of 5-day orders with the same coefficients value $\widetilde{C_w} = 0.4$ *and* $\gamma = 2$ in the Figure 5.12 (the detailed simulation results of the 4 designs are in the appendix). These graphs have the similar pattern as the design #1 graph, the optimal picker schedules of order design #2, #3, and #4 are also from 9 AM to 7 PM.

The difference among the 5 design of orders is the item frequency of orders. The simulation results of back-stock strategy show the back-stock strategy is more effective on the orders with higher item frequency. As item frequency of orders higher, the picking time for each order is less, which make the order waiting cost and the fulfillment cost decrease in general. Due to the order waiting time decrease among designs, the graphs show the fulfillment cost gaps among the picker schedule length become smaller, and a tendency of later picker start time. In the design #5, the optimal picker schedule moves to a 9-hour length from 10 AM to 7 PM. But the fulfillment cost of the optimal schedule and schedule 9 AM to 7 PM or 10 AM to 8 PM is close, which means the high item frequency orders give more flexibility in the picker scheduling.

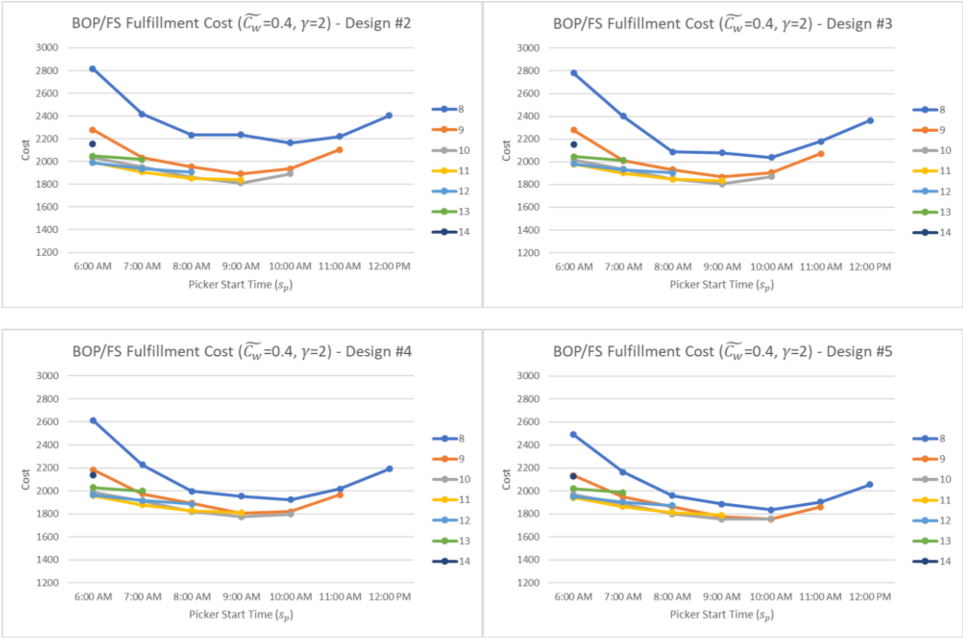


Figure 5.12 BOPS fulfillment cost (waiting time cost coefficient 0.4, overnight waiting 2 hours) in designs.

5.4.3.3 Cost Coefficients. The delivery tardiness (waiting) cost is balanced with the picker schedule cost. We show the graphs of design #1 as the example, as the waiting time cost coefficient ($\widetilde{C_w}$) decreases, the picker schedule length dominates the fulfillment cost. Comparing to $\widetilde{C_w} = 0.4$, when $\widetilde{C_w}$ decreases to 0.2, the optimal schedule shortens to the 9-hour with the same start time. As $\widetilde{C_w}$ continues decreasing to 0.1 and 0.05, the 8-hour schedules give the minimum fulfillment cost.

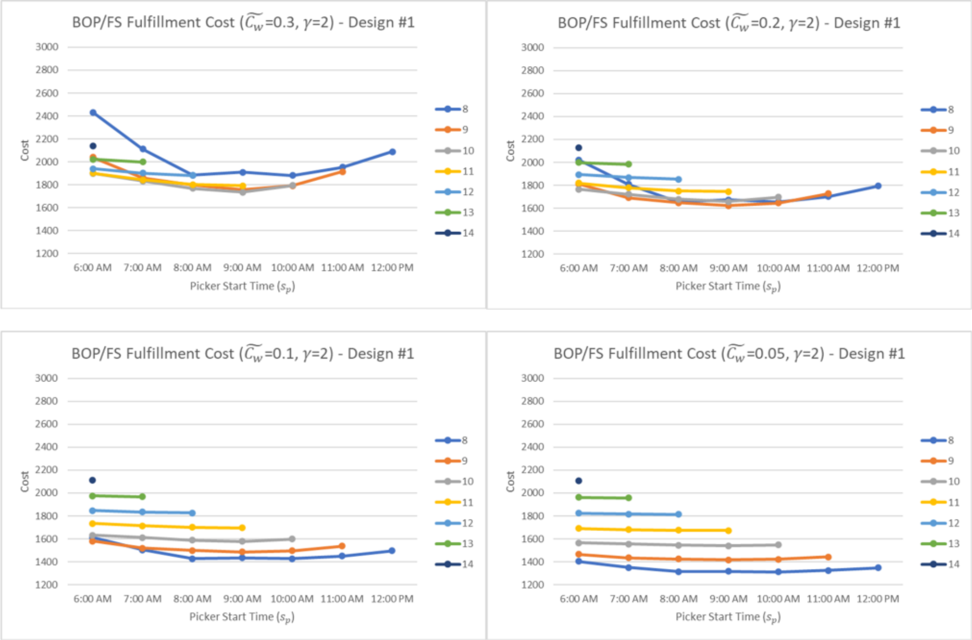


Figure 5.13 BOPS fulfillment cost (overnight waiting 2 hours) of design #1.

For each set of 5-day orders, the order fulfill time ($F_{n,d}$) is decided by the picker schedule. We still take the simulation results of design #1 orders as the example, Table 5.14 shows the number of next-day fulfilled orders under each picker schedule. The schedules end at 8 PM barely leave orders to the next day.

Table 5.14 Number of Next-day Fulfilled Orders – Design #1

Picker Start Time (S_p)		6:00 AM	7:00 AM	8:00 AM	9:00 AM	10:00 AM	11:00 AM	12:00 PM
Daily Labor Scheduled Hours ($E_p - S_p$)	8	475	339	227	168	98	47	5
	9	326	224	158	95	43	0	
	10	224	158	95	43	0		
	11	158	95	43	0			
	12	95	43	0				
	13	43	0					
	14	0						

When we change the overnight hours $\gamma = 1$, it only decreases the overnight waiting cost of the next-day fulfilled orders. With the same waiting time cost coefficient $(\widetilde{C_w})$, the fulfillment cost of schedules having no next-day fulfilled orders stay same. Comparing to the above graphs of $\gamma = 2$, the graphs below ($\gamma = 1$) show the fulfillmnet cost decreases from the left side. There is no change of the 14-hour full coverage schedule. For each schedule length, the schedule with the lastest picker start time also stay same.

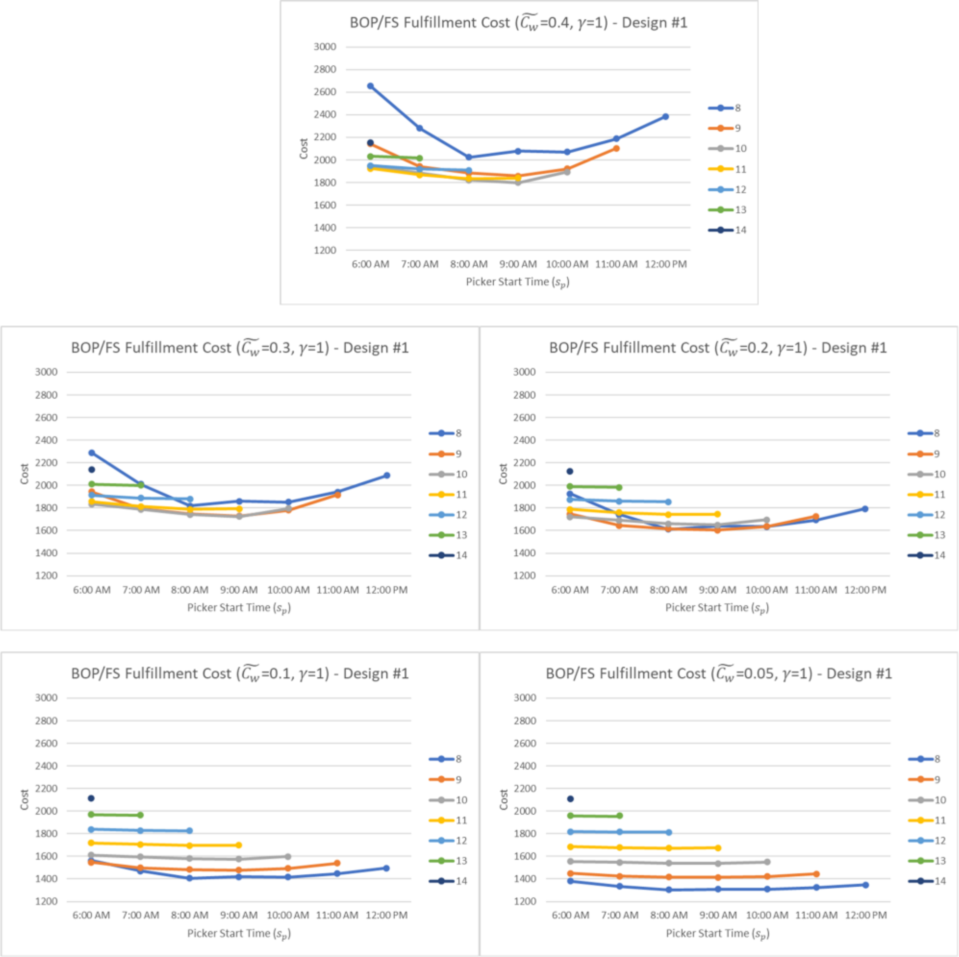


Figure 5.14 BOPS fulfillment cost (overnight waiting 2 hours) of design #1.

CHAPTER 6

SUMMARY AND FUTURE RESEARCH

Fulfillment of online retail orders is a critical challenge for retailers since the legacy infrastructure and control methods are ill suited for online retail. The primary performance goal of online fulfillment is speed or fast fulfillment, requiring received orders to be shipped or ready for pickup within a few hours. Several novel numerical problems characterize fast fulfillment operations and this research solves two such problems, F-Warehouse order consolidation and BOPS store picking problems.

6.1 Summary

Order fulfillment warehouses (F-Warehouses) are a critical component of the physical internet behind online retail supply chains, and typically operate with an explosive storage policy. That is, each item is stocked in multiple random locations dispersed throughout the warehouse. Orders are then picked and collected in totes which are assigned to one of many packaging stations, where items belonging to the same order are consolidated into a shipment package. There is a one-to-many relationship between customer orders and totes. This research formulates and solves the order consolidation problem. At any time, a batch of totes are to be processed through several available order packaging stations. Tote assignment to a station will determine whether an order will be shipped in a single package or multiple packages. Reduced shipping costs are a key operational goal of an online retailer, and the number of packages is a determining factor. The decision variable is which station a tote should be assigned to, and the performance objective is to minimize the number of packages and balance the packaging station workload. This research first formulates the order consolidation problem as an MIP, and then develops two fast heuristics (#1 and #2) plus two clustering algorithm derived solutions. For small problems, the heuristic 2 were on average within 4.1% of the optimal solution, while the heuristic 1, the hierarchical and K-Means clustering algorithms were within 8.5%, 6.7% and 11.6%. For larger problems heuristic #2 outperformed all other algorithms. Performance behavior of heuristic #2 was further studied as a function of size of the tote-order matrix, multi-item order complexity of the tote-order matrix, number of consolidators and the consolidation batch window. A Maxmin ratio indexed in the range number of ordered items minus number of orders was used as a surrogate for solution performance.

S-Strategy fulfillment is a store-based solution for fulfilling online customer orders. Orders are picked from store inventory and then customer picks up from the store (BOPS). A BOPS store has two distinguishing features (i) In addition to shelf stock, the layout includes a space constrained back stock of selected items, and (ii) a set of dedicated pickers who are scheduled to fulfill orders. This research solves two BOPS related problems: (i) Back stock strategy: Assignment of items located in the back stock and (ii) Picker scheduling: Effect of numbers of picker and work hours. For both problems we assume a continuous flow of incoming orders and the objective is fulfillment time and labor cost minimization. We model the store inventory dispersion, order arrival process and order picking process in a BOPS retailer. For the back-stock problem an assignment rule based on order frequency, forward location and order basket correlations achieved a 17.6% improvement over a no back-stock store, while a rule based only on order frequency achieved a 12.4% improvement.

6.2 Future Research

With time and resource limitations, the research on both problems are still with plenty of future research opportunities.

As mentioned in Chapter 4, the twinning orders in a tote-order matrix is a preliminary design for future research. The intention of this design is to link the order consolidation problem with the picklist assignment problem, which is the efficient picking lists generation from the set of pending orders. A picklist of items is collected in a tote, which means the picklist decides which orders in a tote. And a batch of totes form the tote-order matrix in the decision model. With the online order fulfillment in F-Warehouses, the research team has established four decision models with the operation work flows. As a future study, a combination of picking and consolidation algorithms could achieve improvement in both fulfillment time and cost minimization.

With the BOPS store picking problem, the current derivation of order delivery waiting cost considers every order needs immediate fulfillment. In the future study, orders with different due date/time priority could be assigned to different fulfillment groups. For the picking scheduling problem, this research focus on the daily labor hours and start time of one picker. When the model involves multiple pickers, the number of pickers will be a critical decision variable. Combined with the order priority, the picker scheduling problem will show more complexity.

REFERENCES

Acimovic, J., & Graves, S. C. (2015). Making Better Fulfillment Decisions on the Fly in an Online Retail Environment. *Manufacturing & Service Operations Management, 17*(1), 34-51. doi:10.1287/msom.2014.0505

Acimovic, J. A. (2012). Lowering outbound shipping costs in an online retail environment by making better fulfillment and replenishment decisions. *Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.* http://hdl.handle.net/1721.1/77825

Armant, V., Cauwer, M. D., Brown, K. N., & O'Sullivan, B. (2018). Semi-online task assignment policies for workload consolidation in cloud computing systems. *Future Generation Computer Systems,* 82, 89-103. doi:10.1016/j.future.2017.12.035

Berardi, V. L., Zhang, G., & Offodile, O. F. (1999). A mathematical programming approach to evaluating alternative machine clusters in cellular manufacturing. *International Journal of Production Economics, 58*(3), 253-264. doi:10.1016/s0925-5273(98)00211-4

Brusco, M. J., Köhn, H. F., & Steinley, D. (2012). Exact and approximate methods for a one-dimensional minimax bin-packing problem. *Annals of Operations Research, 206*(1), 611-626. doi:10.1007/s10479-012-1175-5

Brusco, M. J., Thompson, G. M., & Jacobs, L. W. (1997). A morph-based simulated annealing heuristic for a modified bin-packing problem. *Journal of the Operational Research Society, 48*(4), 433-439. doi:10.1038/sj.jors.2600356

Cambazard, H., Mehta, D., O'Sullivan, B., & Simonis, H. (2013). Bin Packing with Linear Usage Costs – An Application to Energy Management in Data Centres. *Lecture Notes in Computer Science Principles and Practice of Constraint Programming,* 47-62. doi:10.1007/978-3-642-40627-0_7

Chen, A. (2017). Large-scale optimization in online-retail inventory management. *Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.* http://hdl.handle.net/1721.1/111854

Coffman, E. G., Csirik, J., Galambos, G., Martello, S., & Vigo, D. (2013). Bin Packing Approximation Algorithms: Survey and Classification. *Handbook of Combinatorial Optimization,* 455-531. doi:10.1007/978-1-4419-7997-1_35

Costa, A., Ng, T., & Foo, L. (2017). Complete mixed integer linear programming formulations for modularity density based clustering. *Discrete Optimization, 25*, 141-158. doi:10.1016/j.disopt.2017.03.002

Czyzyk, J., Mesnier, M., & More, J. (1998). The NEOS Server. *IEEE Computational Science and Engineering, 5*(3), 68-75. doi:10.1109/99.714603

Dolan, E. D. (2001). NEOS server 4.0 administrative guide. doi:10.2172/822567

Fourer, R. (2013). Algebraic Modeling Languages for Optimization. *Encyclopedia of Operations Research and Management Science*, 43-51. doi:10.1007/978-1-4419-1153-7_25

Fourer, R., Gay, D. M., & Kernighan, B. W. (1990). A Modeling Language for Mathematical Programming. *Management Science*, *36*(5), 519-554. doi:10.1287/mnsc.36.5.519

Friesen, D. K., & Langston, M. A. (1986). Variable Sized Bin Packing. *SIAM Journal on Computing, 15*(1), 222-230. doi:10.1137/0215016

Gay, D. M. (2015). The AMPL Modeling Language: An Aid to Formulating and Solving Optimization Problems. *Numerical Analysis and Optimization Springer Proceedings in Mathematics & Statistics,* 95-116. doi:10.1007/978-3-319-17689-5_5

Gropp, W. & Moré, J. J. 1997. Optimization Environments and the NEOS Server. *Approximation Theory and Optimization*, 167-182.

Hall, N. G., Ghosh, S., Kankey, R. D., Narasimhan, S., & Rhee, W. T. (1988). Bin packing problems in one dimension: Heuristic solutions and confidence intervals. *Computers & Operations Research, 15*(2), 171-177. doi:10.1016/0305-0548(88)90009-3

Hassin, R., & Rubinstein, S. (2006). An improved approximation algorithm for the metric maximum clustering problem with given cluster sizes. *Information Processing Letters, 98*(3), 92-95. doi:10.1016/j.ipl.2005.12.002

Huang, Z. (1998). Extensions to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values. *Data Mining and Knowledge Discovery,* 2(3), 283-304. doi:10.1023/a:1009769707641

Klein, G., & Aronson, J. E. (1991). Optimal clustering: A model and method. *Naval Research Logistics, 38*(3), 447-461. doi:10.1002/1520-6750(199106)38:33.0.co;2-0

Luna, A. (2015). Characterizing and improving the service level agreement at Amazon. *Master Thesis, Massachusetts Institute of Technology, Cambridge, MA.* http://hdl.handle.net/1721.1/99011

Maiza, M., Labed, A., & Radjef, M. S. (2012). Efficient algorithms for the offline variable sized bin-packing problem. *Journal of Global Optimization, 57*(3), 1025-1038. doi:10.1007/s10898-012-9989-x

Mao, F., Blanco, E., Fu, M., Jain, R., Gupta, A., Mancel, S., Yuan, R., Guo, S., Kumar, S., Tian, Y (2017). Small Boxes Big Data: A Deep Learning Approach to Optimize Variable Sized Bin Packing. *2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService).* doi:10.1109/bigdataservice.2017.18

Mason, A. J. (2013). SolverStudio: A New Tool for Better Optimisation and Simulation Modelling in Excel. *INFORMS Transactions on Education, 14*(1), 45-52. doi:10.1287/ited.2013.0112

Onal, S., Zhang, J., & Das, S. (2017). Modelling and performance evaluation of explosive storage policies in internet fulfilment warehouses. *International Journal of Production Research, 55*(20), 5902-5915. doi:10.1080/00207543.2017.1304663

Onal, S., Zhang, J., & Das, S. (2018). Product flows and decision models in Internet fulfillment warehouses. *Production Planning & Control, 29*(10), 791-801. doi:10.1080/09537287.2018.1469800

Rao, M. (1971). Cluster Analysis and Mathematical Programming. *Journal of the American Statistical Association,* 66(335), 622-626. doi:10.2307/2283542

Rao, R., & Iyengar, S. (1994). Bin-packing by simulated annealing. *Computers & Mathematics with Applications, 27*(5), 71-82. doi:10.1016/0898-1221(94)90077-9

Song, W., Xiao, Z., Chen, Q., & Luo, H. (2014). Adaptive Resource Provisioning for the Cloud Using Online Bin Packing. *IEEE Transactions on Computers, 63*(11), 2647-2660. doi:10.1109/tc.2013.148

Vinod, H. D. (1969). Integer Programming and the Theory of Grouping. *Journal of the American Statistical Association, 64*(326), 506-519. doi:10.1080/01621459.1969.10500990

Xu, P. J. (2005). Order fulfillment in online retailing: what goes where. *Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA.* http://hdl.handle.net/1721.1/33672

Xu, P. J., Allgor, R., & Graves, S. C. (2009). Benefits of Reevaluating Real-Time Order Fulfillment Decisions. *Manufacturing & Service Operations Management, 11*(2), 340-355. doi:10.1287/msom.1080.0222