University of Louisville
ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

12-2016

Order picking strategies for healthcare warehouses.

Ehsan Khodabandeh
University of Louisville

Follow this and additional works at: https://ir.library.louisville.edu/etd

 Part of the Industrial Engineering Commons, and the Operational Research Commons

Recommended Citation
Khodabandeh, Ehsan, "Order picking strategies for healthcare warehouses." (2016). *Electronic Theses and Dissertations.* Paper 2601.
https://doi.org/10.18297/etd/2601

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

ORDER PICKING STRATEGIES FOR HEALTHCARE WAREHOUSES

By

Ehsan Khodabandeh
B.S., Iran University of Science & Technology, 2007
M.S., Sharif University of Technology, 2011

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy in Industrial Engineering

Department of Industrial Engineering
University of Louisville
Louisville, Kentucky

December 2016

ORDER PICKING STRATEGIES FOR HEALTHCARE WAREHOUSES

By

Ehsan Khodabandeh
B.S., Iran University of Science & Technology, 2007
M.S., Sharif University of Technology, 2011

A Dissertation Approved On

November 22, 2016

by the following Dissertation Committee:

Gail DePuy, Dissertation Director

Ki-Hwan Bae, Dissertation Committee Member

Gerald Evans, Dissertation Committee Member

Gerold Willing, Dissertation Committee Member

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor and mentor, Dr. Gail DePuy. Her guidance, patience, and support encouraged and helped me immensely throughout this research. I could not ask for a better adviser and friend.

I would also like to thank Dimitrios Drosos for his time and support of this work by providing invaluable data and insight. His ideas steered this research to the right direction.

ABSTRACT

ORDER PICKING STRATEGIES FOR HEALTHCARE WAREHOUSES

Ehsan Khodabandeh

November 22, 2016

Order picking is the process of collecting goods and items in specified quantities from storage locations, in response to customer orders. Since many labor resources are involved in this process, finding ways to make it more efficient have been a primary goal for researchers and practitioners. Determining a better allocation of products to the storage areas, finding the best route and sequence to pick multiple products, and choosing the best picking policies to minimize congestion in the aisles are just a few of many objectives regarding order picking process.

Due to regulatory compliances and the chance of product spoilage, additional criteria should be taken into account when dealing with order picking in a healthcare warehouse. Following the first-expired, first-out (FEFO) principle and fulfilling an order from a single manufacturing batch, are two of the requirements that are considered in this research. Moreover, minimizing the traveled distance to picking locations, minimizing the penalty or cost of using the space by depleting it quicker, and maximizing the probability of a successful pick in the future by considering the order size distribution are the other objectives that have been studied in this research.

The desired order picking problem is formulated and solved as a multi-objective mixed integer programming optimization model. Assigning importance weights to each objective and obtaining one single solution does not provide a full picture of all possible and potentially attractive solutions. On the other hand, providing all the

solutions is not always achievable as it is computationally expensive and most of the time a set of rules and regulations drive decision makers toward the chosen solution. Finally, this research focuses on solution simplicity, generality, and practicality since the solution will be implemented by order pickers.

To achieve this goal, a novel approach using association rule mining is presented. Products are classified in different groups and some order picking rules are derived based on the relations of these classes together. These rules are then compared with the results of multi-objective optimization model to evaluate their quality. The comparison results showed that surprisingly, some simple rules extracted from the preferences of the decision maker, can obtain good quality solutions. For example, in products with high order variability, it is possible to generate solutions within an average gap of less than $\pm 8.8\%$ from the multi-objective optimization solutions.

The findings of this research leads to the following primary recommendations:

- Orders should be picked based on the expiration dates of the batches (FEFO principle) unless, the amount of inventory in the location is exactly equal to the order amount. In that case, regardless of the expiration date and the traveled distance to the location, it is better to pick from that location and make the space available.

- If product replenishment is not instantaneous, regardless of the expiration date and the traveled distance to the location, it is recommended to **not** pick from a location which can lead to an undesired and low amount of inventory. Due to order variability of the product, it is better to wait until the inventory in that location transfers to a desirable level.

- If fast replenishment of products is obtainable, then prioritizing traveled distance over the objectives linked to space usage and inventory is advisable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I
INTRODUCTION

A. Introduction

According to *Eighth UPS Pain in the Chain Survey* 2015, the core supply chain issues for healthcare logistics decision makers are: 1) product security; 2) regulatory compliance; 3) product damage and spoilage and, 4) managing logistics, warehousing, and transportation costs where the latter is a continuing management struggle. A big part of the warehousing costs relate to order picking problem.

Order picking, the process of retrieving items from storage in response to a customer request, is the most labor-intensive and costly activity in almost every warehouse comprising approximately 55% of the total warehouse operating expenses (Tompkins et al. 2010). This is true for healthcare warehouses as well. A report by United Parcel Service (UPS) showed that picking and order fulfillment consumes 54% of the cost in the average healthcare distribution center (*Best practices for managing cost in the healthcare supply chain*).

Order picking systems are classified as shown in Figure. 1. The most common system, which is used in this research, is picker-to-parts, where the order picker walks or drives along the picking locations to pick items (De Koster 2004). The major design issues in a warehouse such as the shape and size of the warehouse, and the number of pickers or vehicles are not of a concern here. The focus of this research is on a set of stock keeping unit (SKU)-related criteria for order picking in a picker-to-parts system. These criteria are location, expiration date, and stock on hand of each SKU.

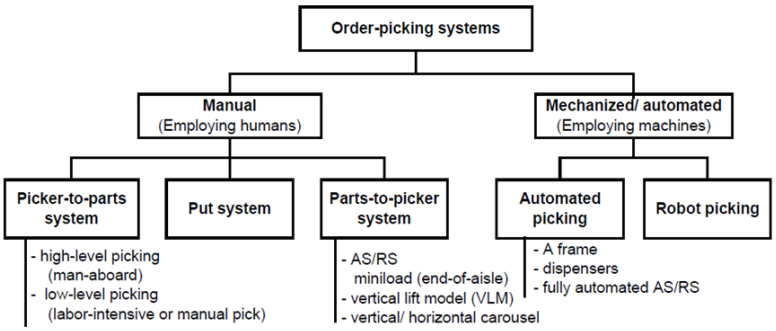


FIGURE 1 – Classification of order picking systems (De Koster 2004)

Note that SKU, refers to a specific item stored in a specific location and is considered the most disaggregated level when dealing with inventory. It is possible to have multiple SKUs for the same item. Throughout this research, the term *batch* is used to distinguish between SKUs of an item with different expiration date.

When receiving an order, a picker, either automated or manual, is dispatched to pick the appropriate quantity of each item from its storage location and bring them back to the shipping area. Finding a way to minimize the distance or time traveled by the picker is a common objective. As observed in a survey by Gu, Goetschalckx, and McGinnis 2007, 38% of warehouse operation planning problems are related to routing problems. But how are the storage locations selected for a pickup?

If an item type is stored in only one location, then there is no difficulty determining from which storage location to pick. However, if an item type is stored in more than one location, the decision must be made as to which location to pick when filling an order. Moreover, different SKUs of perishable items might be stored in different locations as is the case in warehouses storing healthcare items. Due to food and drug administration (FDA) regulations for the pharmaceutical and healthcare industry, an item is retrieved based on the first-expired, first-out (FEFO) principle. This means that for an item, SKUs with the nearest expiration dates should be picked from stock first.

Also as part of the regulatory requirements, an order for a healthcare item

must be filled from a single manufacturing batch. Imagine an item is found to be defective or spoiled. When all the units in an order are picked from a single batch, it is easier to track down the source, investigate the causes, and remove the faulty items from public consumption.

The storage locations for an item are not all within the same distance from the shipping area and the SKUs, have different features. Besides, picking decisions become more complicated by the fact that the sizes of future orders are uncertain. Since this uncertainty, the restriction on expiration date, and also fulfilling an entire order from a single batch will require more precise analysis, it adds to the complexity of the decision making process.

Considering the stochastic nature of future demands and having different batches of an item in multiple locations, one needs to decide from which location to pick an order to optimize the desired objective. The following are a few of the various objectives often taken into consideration in order picking optimization:

- Minimizing the order picking time
- Minimizing the overall throughput time
- Minimizing travel distance
- Minimizing total costs
- Maximizing the expected number of orders that can be completed per unit time or equivalently, minimizing order cycle time (Jewkes, Lee, and Vickson 2004)
- Maximizing the order fill rate (Rim and Park 2008)
- Maximizing throughput rate (Pan and Shih 2008)

Examining other objectives such as minimizing the waste (i.e. amount left in each location which might not be used for future picks) or maximizing the space

utilization by consuming batches completely and leaving no remainder in each location, can be very interesting. Additionally, studying multiple objectives can make the problem more realistic and closer to the situations happening in the healthcare warehouses.

In this research, order picking with regard to certain constraints and multiple objectives in a healthcare and medical warehouse is investigated. Minimizing travel distance, minimizing waste, and maximizing space utilization along with taking into account the FEFO principle and uncertainty of order sizes, form the structure of objective functions. To calculate the picker's travel distance, the distance from each batch's storage location to input/output (I/O) point is used. To minimize waste and maximize space utilization, quantity of an item in different locations and also information to estimate future demands such as distribution function for each item's order sizes are utilized. Finally, including each batch's expiration date in the objective function and trying to choose the nearest expiry one, is applying the FEFO principle in order picking decision making process.

The goal of this research is to find *simple, general*, and *practical* order picking strategies for healthcare warehouses. In simple terms these strategies will be in forms of:

"If situation A happens, then pick based on strategy S_1; if situation B, C, and D happen, then pick based on S_2; etc."

So, these situations and strategies are sets of *"If ... Then ... "* scenarios such as the examples below:

- For *high demand* items, if order size is X_1, then pick from the *closest location* available.
- For *high demand* items if order size is X_2, then pick from the location with the *highest stock*.

- For items with *low variability demand*, first pick from the batch in a location which *expires first*. In case of having one batch in multiple locations or all batches in the *acceptable range of expiration dates*, then pick from a location with *average stock*.
- For *non-frequent items with high variance*, pick from the location with *least stock*.
- ...

To achieve the goal of creating these rules, one needs to i) define the situations, ii) determine meaningful strategies, and iii) find out ways to establish a link between them. But how to determine the value of these rules and how *good* they are? How to determine good and valuable? What are the metrics to measure such values? Is there any reference or method that can be used for determining the goodness of the rules? If so, what are those methods and how the comparison is made? How to model this problem to be eligible for such comparisons?

By starting from the goal of creating order picking strategies, the problem was broken down into series of questions that all will be addressed in this research.

B. Literature Review

Order picking has been the topic of many researches over the past few decades. Figure. 2 shows how order picking is classified and where it is placed among the warehouse operation planning problems (Gu, Goetschalckx, and McGinnis 2007). While there has been much research on the general order picking problem, little research regarding order picking for healthcare warehouses was found. The readers may refer to Gu, Goetschalckx, and McGinnis 2010, 2007 for detailed reviews on warehouse operation, design, and performance evaluation and also, comprehensive review of De Koster, Le-Duc, and Roodbergen 2007 on design and control of manual

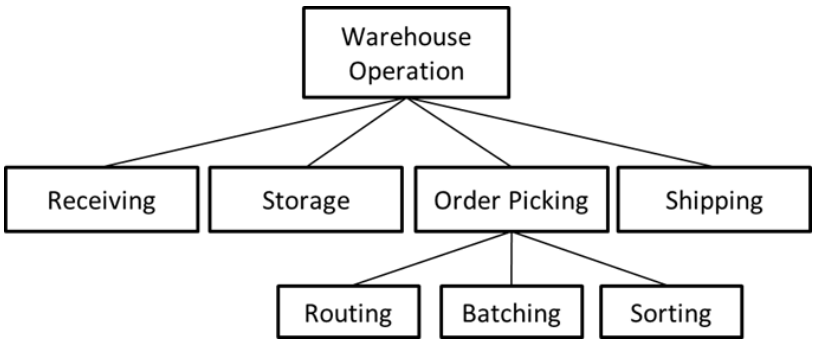


FIGURE 2 – Warehouse operation problems framework

order picking processes.

The literature is rich particularly in two of these warehousing functions; storage location assignment problem (SLAP) and routing and sequencing decision in order picking operations. SLAP determines where to store items in order to reduce material handling cost. Routing and sequencing, on the other hand, determines the best sequence and route of the locations for storing and picking a set of items. Gu, Goetschalckx, and McGinnis 2007 observed that 32% and 38% of their total surveyed literature on warehouse operation planning problems were about SLAP and routing problems, respectively.

As mentioned above, the goal of solving SLAP is to find the best locations for storing items in order to reduce material handling cost. Minimizing material handling cost or equivalently, traveling cost, traveling distance, or traveling time is one of the objectives which researchers have strongly focused on to improve order picking efficiency. Tompkins et al. 2010 showed that in fact, half of the order picker's time is spent on traveling (Figure. 3). So, it is not surprising to see most of the researches are dedicated to improving this dominant activity.

However, as mentioned before, one overlooked decision is how the picking locations are selected in case of existing multiple storage locations. The core of this research is in studying the effects and interactions of several factors on selecting the pick location in healthcare warehouses. In this work, the factors, which are relevant

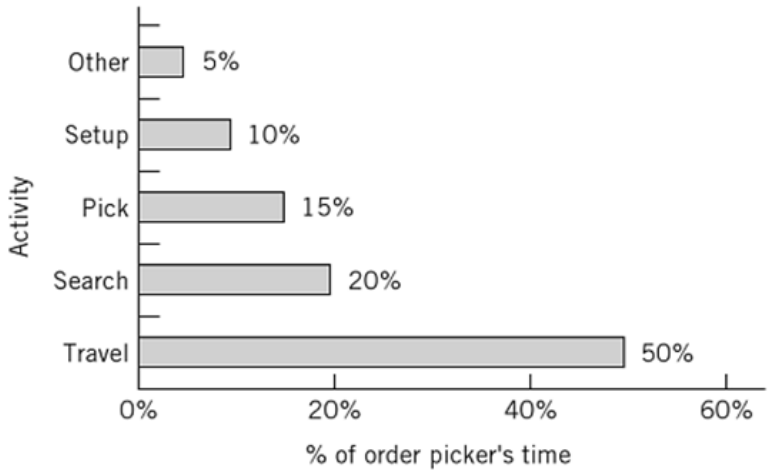


FIGURE 3 – Typical distribution of an order picker's time (Tompkins et al. 2010)

to the location of an item, are distance from the I/O, expiration date of items, and the quantity of items in that location. Considering expiration date of items in order picking decisions is a rather new topic which has just been seen in a patent by Klots et al. 2003.

Klots et al. 2003 created a set of rules for an order picking system to maximize throughput and reduce carrying costs by reducing the number of stops in the process of fulfilling an order. In their method, if the system found multiple locations that stock the same item, it chose the location with sooner expiration date. If the expiration dates were within the same period, then the system selected the location with the fewest units of items hoping to create more available space. In case that no location had sufficient quantity of an item, the order might be picked collectively over two or more locations. This is one of the main differences of their work with this research since order splitting is conditional in this research. So, order can be picked from multiple locations as long as it is from the same batch.

As mentioned above, studying the effects of several factors simultaneously on order picking performance is of this research interest. Regarding that matter, it is seen that in the last decade, researchers began jointly considering multiple warehousing

functions and studying their interactions and effects on order picking operations. The studies conducted by Jewkes, Lee, and Vickson 2004, Petersen and Aase 2004, Chan and Chan 2011, Pan and Shih 2008, and Molnar and Lipovszki 2002 are just to name a few, yet, none of them are specifically related to healthcare warehouses.

Jewkes, Lee, and Vickson 2004 determined product location, picker home base location, and allocation of products to pickers concurrently in a multiple-picker order picking line, to minimize the expected order cycle time. Each picker operated from a fixed home base on a pick-to-order line in which a picker worked on one order at a time. The authors determined the optimal policy by using a greedy approach. They also developed a dynamic programming algorithm to determine the optimal product allocation and picker locations in case of fixed product locations.

Petersen and Aase 2004 examined several picking, storing, and routing policies simultaneously on a manual bin-shelving order picking operation. They used simulation to determine which policy or combination of policies has the largest impact on minimizing the total pick time for all the orders in a day. Results of the simulation showed that batching has the greatest impact on reducing the pick time, particularly for smaller order sizes. The results also showed that although class-based and volume-based storage policies may increase picker congestion within aisles containing the most popular items, they performed better than random policy in terms of the amount of picker's travel time.

Chan and Chan 2011 used simulation to compare the effects of different storage assignment policies, routing policies, and pick densities on the overall performance of order picking in a manual-pick multi-level rack warehouse. The performance of order picking was measured in terms of order retrieval time and travel distance. The results of simulation with ARENA showed that different combination of factors performed differently under different performance indicators. The one performed better in terms of order retrieval time resulted in longer travel distance. With regards to the total

travel distance, the performance of storage assignment policy changed due to an increase in pick density. Also, they found that the use of storage assignment policy was dependent on the type of the storage system.

Pan and Shih 2008 studied an order picking system with multiple pickers and parallel-narrow-aisles in a picker-to-parts warehouse. They chose the throughput rate as the performance measure for order picking operation and modeled the problem as an open restricted queueing network. The authors used an approximation method to calculate the throughput rate and validated the results with FlexSim simulator output. They considered total travel time and congestion effect simultaneously and by using simulation, they compared the efficiency of different storage assignment policies.

Molnar and Lipovszki 2002 solved a multi-objective routing and scheduling of order pickers in a warehouse. The objectives were creating well-built loads on the shortest possible route. Using genetic algorithm they evaluated a number of scenarios and determined the number of order pickers per shift and the best sequence of releasing the pick lists to be retrieved.

As mentioned earlier, the literature regarding order picking for healthcare warehouses is scarce. The work of Jorge et al. 2012 was the only one found on order picking operations in a picker-to-parts pharmaceutical warehouse. They used an agent-based model of the warehouse to simulate the activities and operations in NetLogo modeling platform. The purpose of their simulation was trifold; to assign the correct number of pickers for the average number of served orders, to calculate the orders arrival rate, and to provide a tool to analyze the performance of order picking process.

In a different path from other researchers, Rim and Park 2008 divided order picking into two general stages, planning and execution. Planning is the stage where it is decided which set of orders to pick among the received orders; and the execution

stage focuses on trying to pick efficiently to minimize travel distance, order picking time, etc. The vast majority of research targets the execution stage. Rim and Park 2008 was the only work found to address the planning stage.

As the pioneers of researching the order picking planning problem, Rim and Park 2008 studied a situation where some orders may not be fulfilled due to the shortage of inventory. An order is fulfilled when there are enough inventories for satisfying all the items in that order. They determined which set of orders to pick in order to maximize the order fill rate. They formulated the problem as a linear programming (LP) problem and compared its performance with that of the first-come, first-served (FCFS) rule. Simulation results showed that even though using FCFS rule reduced the number of required pickers, the authors' proposed LP approach, outperformed FCFS in order fill rate which resulted in higher customer satisfaction.

Because of incorporating expiration date cost as one of the objective functions, it is worth mentioning the research of Poulos et al. 2001. They developed a Pareto-optimal genetic algorithm for solving multi-objective warehouse replenishment problem. The objectives were the distance from the delivery points, the distance from the collection points, the distance from the picking locations, the expiration date of the product and its seasonal demand. To incorporate the preference of the decision maker, the authors assigned importance weights to each attribute of the objective function.

In the reviewed papers and in the literature, generally, it is assumed that all the available inventory of an item is identical and there is no difference between the inventories of an item in different locations. In reality, however, this is not always true. One item in two different locations can have different expiration dates (like the cases of Klots et al. 2003 and Poulos et al. 2001) or batch numbers.

As explained in the introduction section, studying multi-objective order picking system in a healthcare and medical warehouse is of this research interest. Also, to

the best of authors' knowledge, this is the first time that differences between batches, i.e. expiration date, have been considered in optimizing the order picking operation. Furthermore, it is very common to solve multi-objective optimizations by assigning importance weights to each objective which is not the case of this research. Because doing so, will result in one solution and cannot provide a full picture of all the possible solutions. This matter will be discussed in more detail in Chapter III.

It is assumed that each order is for a single type of item and every time an order is received, all the items must be picked only from one batch. Therefore, the order is allowed to be split between multiple locations but not between multiple batches. In most previous studies on the order picking problems the order sizes were known. However, it is assumed here that there is no information available on size of the order, thus, they should be treated stochastically. Moreover, the knowledge about the stochastic nature of the orders is utilized to deal with satisfying the future orders and then, this information is used for determining the order picking location.

In summary, this research is different from the existing literature in several aspects:

- Using multi-objective functions for deciding on the picking location
- Employing stochastic order sizes
- Considering expiration date in order picking decisions
- Considering the uniqueness of the batch in picking decision (no order splitting between batches)
- Considering the probability of future order sizes in order picking operation
- Using a novel decision making approach for finding the relation between multi-objective order picking optimization and order picking strategies (Chapter IV)

The rest of this research is organized as follows. Chapter II discusses the problem in more detail and introduces the multi-objective formulation for the order picking problem. Chapter III goes into details of multi-objective optimization and methods that are used in handling the desired problem. The solution structure and ways to compare different solutions are explained. Chapter IV discusses the novel approach for finding the relation between solutions in multi-objective optimization and structuring the order picking strategies. Chapter V applies all the findings on sample datasets and discusses the results. At the end, conclusions and the path for future works are explained.

CHAPTER II
PROBLEM DESCRIPTION

A. General Description

Consider a warehouse or a distribution center which holds and supplies many items to many customers. When customers place orders, they are picked from the available inventory and shipped to the customers. It is assumed that the items will not be depleted during the pick cycle, i.e. there are enough inventories to cover the orders. It is also assumed that the replenishment is instantaneous and as soon as a storage location is empty, the replenishment happens. In other words, replenishment follows the reorder point, order-up-to-level (s, S) policy with $s = 0$.

Usually in order picking, demand has been treated as given or known in advance. In this research, instead, it is assumed that the demand for each item is unknown. However the distribution function for each item's order sizes can be determined using previous order sizes.

Each type of item can occupy more than one location and the inventory of an item in each location can be any non-negative number. Also the same item in different locations can be from the same batch or different batches, i.e. a batch can occupy more than one location. Order splitting among batches is not allowed. Therefore, any time an order is received it has to be fully satisfied from one batch. However, one can pick from multiple locations as long as the batches are the same. This work addresses only *single order picking* operation in the warehouse when the order picker picks one order at the time.

There are four criteria in selecting a location in case of having multiple candidate locations for picking an order. In such a situation, the batch with sooner expiration date, bigger remainder (remaining stock after an order pick), closer location to I/O, which also can become totally depleted is preferred. The fourth criterion is whether the location becomes empty after picking an order. So, among several candidate locations for an order pick, the one that can be completely empty after the pick should be the preferred one. In other words, priority should be given to a location where its stock is equal to the order size.

Preferring a location which can become completely empty after a pick seems reasonable, because it increases the amount of available empty space and decreases inventory holding costs. But it contradicts the rule of giving a higher priority to a location with bigger remainder, which is a counterintuitive rule. The reason behind this rule is explained with an example.

Suppose, there are two locations holding item "A", location one with an inventory of 12 units and location two with an inventory of 17 units. As mentioned before, in this research it is assumed that the order size is unknown and just its distribution function is known. Suppose that order size for item "A" follows a normal distribution with mean of 8 and standard deviation of 2. Now, an order of size 8 is received and it is possible to pick this order from either of the two locations. If the order is picked from the first location with 12 available units of item "A", 4 units will remain and if the pick happens from the second location with 17 units of inventory, 9 units will remain. Based on the demand distribution function of item "A", the chances of receiving an order of size 4 and smaller in case of picking the order from location one or receiving an order of size 9 and smaller in case of picking the order from location two, are:

Probability of receiving an order of size 4 and smaller $= F(4) = \Phi(\frac{4-8}{2}) = 0.0228$

Probability of receiving an order of size 9 and smaller $= F(9) = \Phi(\frac{9-8}{2}) = 0.6915$

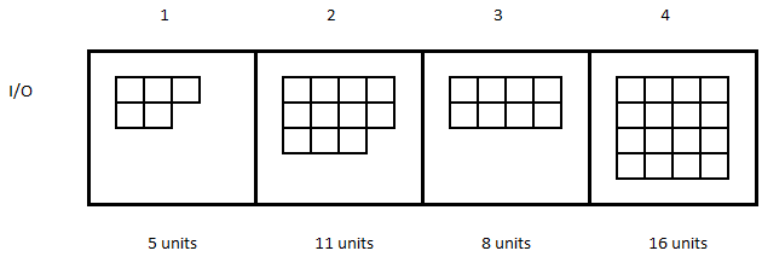


FIGURE 4 – Illustration of the order picking rules

It is seen that if the order is picked from the first location and 4 units are left in the inventory, the chances of having an order for 4 and fewer units in future is less than 3% while by picking the order from the second location, 9 units are left in the inventory which can satisfy another order of item "A" in the future with more than 69% chance. This is why selecting a location with bigger remainder after a pick has given a higher priority.

Now the question is how much each of those four criteria should be weighted. There is an importance weight associated with each of them. One analyst might consider expiration date as the most important criterion and gives it a higher weight while another decision maker might prioritize the picking rules differently. The following example illustrates how these order picking criteria are evaluated in a hypothetical situation.

Suppose there is an order of size 8 for an item which is stored in four different locations as shown in Figure. 4. Without loss of generality it is assumed that the expiration dates for these four batches are in the acceptable range. Also, it is assumed that the importance weights associated to each of the order picking criteria are in favor of complete depletion, higher remainder, sooner expiration date, and closer to the I/O, respectively. So, for instance, the location with higher remainder is preferred over the one closer to the I/O.

Location 1 does not have enough inventory to satisfy this order. The potential remainder of locations 2, 3, and 4, in case the order is picked from them, is 3, 0, and

8, respectively. Location 3 has a remainder of zero which means it can be completely empty and therefore it has the highest priority for order picking. After that, although location 2 is closer to the I/O, but location 4 has a higher remainder which makes it the second favorite location and finally location 2 becomes the third candidate location for picking an order of size 8.

Note that, an analyst who favors closer locations more than the other criteria, has a totally different solution for the same example (in that situation, locations 2, 3, and 4 will be the first, second, and third candidates, respectively). The conflicting nature of these four criteria was observed in the above-mentioned example. As a higher remainder favors location 4, complete depletion favors location 3, and closer distance favors location 2 over others for picking an order of size 8. This contradiction makes it necessary to use a multi-objective approach to solve the explained order picking problem.

Therefore, the desired problem is categorized as a multiple-criteria decision making (MCDM) problem. Typically, there is not a unique optimal solution for such problems and the use of decision maker's expertise in weighing the criteria and differentiating between solutions is inevitable.

B. Problem Formulation

Here the mathematical model for the order picking problem in a warehouse is presented. In formulating the model, the following assumptions are made:

Model Assumptions

- Demands or order sizes for each item are normally distributed
- The initial locations of items are known (no storage assignment is needed)
- Each batch can occupy multiple locations

- Order splitting between batches are not allowed
- Order splitting between locations are allowed
- The warehouse is operated under single order picking operation (one pick per trip).

The following notations are used in formulation of the problem:

Sets

I set of items

J set of batches

K set of locations

L set of orders

Decision Variables

$q_{ijkl} :=$ quantity of item i picked from batch j at location k for the lth order

$\forall i \in I,\ j \in J,\ k \in K,\ l \in L$

$s_{ijkl} :=$ stock of item i in batch j at location k before picking the lth order

$\forall i \in I,\ j \in J,\ k \in K,\ l \in L$

$r_{ijkl} :=$ stock of item i in batch j at location k after picking the lth order

$\forall i \in I,\ j \in J,\ k \in K,\ l \in L$

$$u_{ijl} := \begin{cases} 1 & \text{If } l\text{th order of item } i \text{ is picked from batch } j\ , \\ 0 & \text{Otherwise} \end{cases} \quad \forall i \in I,\, j \in J,\, l \in L$$

$$z_{ijkl} := \begin{cases} 1 & \text{If } r_{ijkl} > 0, \\ 0 & \text{If } r_{ijkl} = 0 \end{cases} \quad \forall i \in I,\, j \in J,\, k \in K,\, l \in L$$

$$y_{ijkl} := \begin{cases} 1 & \text{If } s_{ijkl} > 0, \\ 0 & \text{If } s_{ijkl} = 0 \end{cases} \quad \forall i \in I,\, j \in J,\, k \in K,\, l \in L$$

Parameters

s_{ijk1} := initial stock of item i located in batch j at location k

e_{ij} := expiration date (or acceptance date) of item i located in batch j

o_{il} := size of (number of units in) the lth order of item i

d_k := distance of location k from I/O

$CDF(x)$:= Cumulative Distribution Function of x

M := a very large number

1. Mathematical Model

Minimize:

$$f_1 = \sum_{i\in I}\sum_{j\in J}\sum_{l\in L}\frac{e_{ij}}{\sum_{j\in J}(e_{ij})^+}u_{ijl} \tag{1a}$$

$$f_2 = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\sum_{l\in L}\frac{d_k}{\sum_{k\in K}(d_k)}y_{ijkl} \tag{1b}$$

$$-f_3 = -\left(\sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\frac{CDF(r_{ijkt})}{K^+} + \sum_{i\in I}\sum_{j\in J}\frac{CDF(\sum_{k\in K} r_{ijkt})}{J^+}\right) \tag{1c}$$

$$f_4 = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K}\sum_{l\in L} z_{ijkl} \tag{1d}$$

s.t:

$$q_{ijkl} \le s_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{2}$$

$$\sum_{j\in J} u_{ijl} = 1 \qquad \forall i \in I,\, l \in L \tag{3}$$

$$\sum_{k\in K} q_{ijkl} = o_{il}u_{ijl} \qquad \forall i \in I,\, j \in J,\, l \in L \tag{4}$$

$$r_{ijkl} = s_{ijkl} - q_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{5}$$

$$My_{ijkl} \ge s_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{6}$$

$$y_{ijkl} \le s_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{7}$$

$$Mz_{ijkl} \ge r_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{8}$$

$$z_{ijkl} \le r_{ijkl} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{9}$$

$$s_{ijkl} = s_{ijkl-1} - q_{ijkl-1} \qquad \forall i \in I,\, j \in J,\, k \in K,\, l \in L \tag{10}$$

$$M(u_{ijl} - 1) \le \sum_{k\in K} s_{ijkl} - o_{il} \qquad \forall i \in I,\, j \in J,\, l \in L \tag{11}$$

$$\sum_{j\in J} e_{ij}u_{ijl} \ge 0 \qquad \forall i \in I,\, l \in L \tag{12}$$

$$z_{ijkl},\, y_{ijkl},\, u_{ijl} \in \{0,1\}; \qquad s_{ijkl},\, q_{ijkl},\, r_{ijkl} \ge 0 \text{ and int} \tag{13}$$

The objective function consists of four terms associated with expiration date, distance

from the I/O, item's remainder, and item's total depletion, respectively.

Expiration date's value function (1) calculates the remaining lifespan or age of the item. To normalize this function, the remaining lifespan in a particular batch is simply divided by the summation of all the remaining lifespans of that item in every batch. So, a batch with sooner expiration date holds a smaller portion of this fraction. Note that the positive sign in the denominator is for considering only the not-expired items.

Expiration date function with a negative value, can be considered as an alternative objective function which can be interpreted as the salvage value or selling value of each item at any given point. Since newer items have a higher value, it is preferred that they are kept longer (so the older items should be consumed or picked sooner). The higher the salvage function, the better. So, its negative value can be minimized.

The normalized value function for distance (2), the second term, is formed by dividing the distance from the I/O by the summation of all the distances from the I/O of that item in every location.

The value function for the item's remainder (3) has two parts; location remainder and batch remainder for the final pick, t. Location remainder is the normalized cumulative distribution function of the remainder in each location which describes the probability of receiving an order size of equal or smaller than the remainder. The batch remainder is defined the same but for each batch, since each batch can occupy more than one location. Because a larger remainder is preferred, this term appears with a negative sign in the objective function.

The last term in the objective function (4) is to describe the complete depletion in the model. Because selecting a location that can be totally depleted upon an order pick is preferred, any location with any remainder other than zero will be penalized. This term can also be interpreted as the *space usage* as there is no cost/penalty

associated with the empty space.

The first constraint (2) shows that an item can only be picked from the locations to which it has any stock. Constraint 3 ensures that for each new arriving order, each item can only be picked from one batch. Constraint 4 shows the relation between batches and locations. So if an item is picked from a batch, it can be picked from a single or multiple locations occupied by that batch. Constraint 5 calculates the number of remaining items in each location if an order is picked from that location. Constraints 6 and 7 ensure that if any amount of item is left in a location, that location can still become a candidate location for the future picks, but if no stock is left, that location will not be visited in the future. Constraints 8 and 9 ensure that when the stock of an item can be completely consumed in a location due to an order pick, its depletion variable, z_{ijkl}, obtains zero. Alternatively, the depletion variable of a location that can end up with any positive remainder after an order pick, obtains a value of one to be penalized in the objective function. Constraint 10 ensures that the stock of an item at each location after the pick is updated based on the stock prior to the pick and the quantity picked from that location. Constraint 11 ensures that for each item, there is enough stock in at least one batch to satisfy the order. Constraint 12 makes sure that the items that are picked to satisfy an order are not expired. Constraint 13 states that z_{ijkl}, y_{ijkl} and u_{ijl} are binary variables, s_{ijkl}, q_{ijkl}, and r_{ijkl} are non-negative integers.

The model developed above is a multi-objective integer nonlinear programming (MOINLP) model which is very difficult to solve. The nonlinearity of the problem is due to the CDF function in equation 3 in the objective function. To overcome this nonlinearity, the CDF function is approximated using a piecewise linear representation as shown in Figure. 5. This piecewise approximation is done by introducing SOS2 variables. Special ordered set of type 2 or SOS2, is a set of variables where only two adjacent variables of the set can be non-zero and positive in the solution.

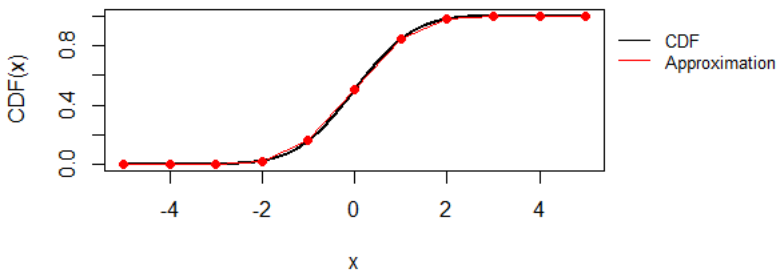


FIGURE 5 – CDF of normal distribution and its piecewise approximation

The piecewise approximation of the nonlinear function is defined by v points. Here, the range of break points for CDF of normal distribution is considered to be $[-5, 5]$. Defining steps of size 0.5, there are $v = 21$ points in this range. Each point is shown by the tuple $(g, f(g))$. A point such as x which is between $g_b \leq x \leq g_{b+1}$, can be written as the convex combination of g_b and g_{b+1}. Taking advantage of the SOS2 variables, the CDF of point x can be approximated by:

$$x = \sum_{b \in B} g_b \lambda_b \tag{14}$$

$$f(x) = \sum_{b \in B} f(g_b) \lambda_b \tag{15}$$

$$\sum_{b \in B} \lambda_b = 1 \tag{16}$$

$$\lambda_b \in \text{SOS2} \tag{17}$$

Although after the approximation, the new model becomes a multi-objective mixed integer linear programming problem which is easier than the MOINLP version of the problem, it is still NP-hard. These types of problems are often handled through the use of heuristic algorithms with the goal of obtaining approximate solutions quickly with sufficient accuracy.

In the next chapter, methods for solving the above-mentioned multi-objective opti-

mization problem is discussed.

CHAPTER III
MULTI-OBJECTIVE OPTIMIZATION

A. Preliminaries

Multiple criteria decision making (MCDM) is one of operations research disciplines that evaluates a set of alternatives or solutions with respect to a set of decision criteria or objectives. One class of MCDM problems is multi-objective mathematical programming (MOMP) problem where the alternatives are not explicitly known. In MOMP, an alternative (solution) can be found by solving a mathematical model. The number of solutions can be infinite (in case of continuous variables) or very large to count (in case of discrete variables).

The mathematical model of the desired order picking problem introduced in the previous chapter, falls into the category of MOMP problems. These types of problems are solved using multi-objective optimization methods. For a general introduction to multi-objective optimization, refer to Ehrgott 2006 and Gandibleux 2006.

Before continuing any further, some of the standard terminology and definitions in multi-objective optimization are described.

A multi-objective optimization problem can be stated as follows:

$$\min_{x \in \mathcal{X}} f(x) := \{f_1(x), \ldots, f_n(x)\} \tag{18}$$

where $n > 1$ and $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of constraints or the *feasible set in the decision space* (the space of feasible solutions). The image $\mathcal{Y}$ of $\mathcal{X}$ under vector-valued function $f = f_1, \ldots, f_n$ represents the *feasible set in the criterion space* (the space of objective

function values). Mathematically speaking:

$$\mathcal{Y} := f(\mathcal{X}) := \{y \in \mathbb{R}^n : y = f(x) \text{ for some } x \in \mathcal{X}\}$$

To distinguish between the two spaces, the elements of the decision space are called *solutions* and elements of the criterion space are referred to as *points*.

Typically, in multi-objective optimization problem, there does not exist a single solution that simultaneously optimizes all objective functions, as the objective functions are conflicting (note that if some objective functions need to be maximized, its negative can be minimized). In this case, the notion of *Pareto optimality* is introduced. A solution is called *Pareto optimal* or *efficient* when none of the objective function values can be improved without degrading at least one other objective value. The relation between Pareto optimal and feasible solutions can be described using *dominance* relation.

Definition III.1. Solution $x_1 \in \mathcal{X}$ is said to *(weakly) dominate* solution $x_2 \in \mathcal{X}$, and is shown as $x_1 \preceq x_2$, if and only if it is not worse than x_2 in all the objectives and is strictly better than x_2 in at least one objective. Mathematically speaking:

$$x_1 \preceq x_2 \iff \begin{cases} f_i(x_1) \leq f_i(x_2) & \forall i \in \{1, 2, \ldots, n\} \\ f_j(x_1) < f_j(x_2) & \exists j \in \{1, 2, \ldots, n\} \end{cases} \tag{19}$$

Also, for x_1 to strongly dominate x_2:

$$x_1 \prec x_2 \iff f_i(x_1) < f_i(x_2) \quad \forall i \in \{1, 2, \ldots, n\} \tag{20}$$

Some properties of dominance relation are as follows:

Reflexive: The dominance relation is *not* reflexive since by definition of dominance, each solution does not dominate itself.

Symmetric: The dominance relation is *not* symmetric because $x_1 \preceq x_2$ does not mean $x_2 \preceq x_1$. But the opposite is true, i.e. if $x_1 \preceq x_2$ then $x_2 \npreceq x_1$.

Transitive: The dominance relation is transitive. So, if $x_1 \preceq x_2$ and $x_2 \preceq x_3$, then $x_1 \preceq x_3$.

Note that if $x_1 \npreceq x_2$, it does not imply that $x_2 \preceq x_1$.

Definition III.2. A feasible solution $x \in \mathcal{X}$ is called *weakly efficient*, if there is no other $x' \in \mathcal{X}$ such that $f_i(x') < f_i(x)$ for $i = 1, \ldots, n$. If x is weakly efficient, then $f(x)$ is called a *weakly nondominated point*.

Definition III.3. A feasible solution $x \in \mathcal{X}$ is called *efficient* or *Pareto optimal*, if there is no other $x' \in \mathcal{X}$ such that $f_i(x') \leq f_i(x)$ for $i = 1, \ldots, n$ and $f(x') \neq f(x)$. If x is efficient, then $f(x)$ is called a *nondominated point*. The set of all efficient solutions $x \in \mathcal{X}$ is denoted by $\mathcal{X}_E$. The set of all nondominated points is denoted by $\mathcal{Y}_N$.

Definition III.4. A feasible solution $x \in \mathcal{X}$ is called *ideal* if it minimizes all the objectives simultaneously. In other words, if $x \in \arg\min_{x \in \mathcal{X}} f_i(x) \quad \forall i = 1, \ldots, n$.

Definition III.5. The point $f^I \in \mathbb{R}^n$ is the *ideal point* if

$$f_i{}^I = \min_{x \in \mathcal{X}} f_i(x) \quad \forall i \in \{1, \ldots, n\}$$

.

Definition III.6. The point $f^N \in \mathbb{R}^n$ is the *nadir point* if

$$f_i{}^N = \max_{x \in \mathcal{X}_E} f_i(x) \quad \forall i \in \{1, \ldots, n\}$$

.

The number of Pareto optimal solutions can be infinite or very large to count. The set of Pareto optimal solutions is called *Pareto optimal set, Pareto frontier, Pareto front,* or *efficient front*. As mentioned in Definition III.3, if x is an efficient solution in the decision space, its image in criterion space is a *nondominated point*. The set of all nondominated points is called *nondominated frontier* or *nondominated front*.

As defined in Definitions III.5 and III.6, nondominated front is bounded by ideal point as an array with the lowerbound of all the objective functions in the entire search space and nadir point as the upperbound of each objective function in the Pareto optimal set.

Definition III.7. A set $\tilde{\mathcal{Y}}_N \subseteq \mathcal{Y}$ is called an *approximation* of the set $\mathcal{Y}_N$ if no point in $\tilde{\mathcal{Y}}_N$ is dominated by any other point in $\tilde{\mathcal{Y}}_N$. In other words, all the points in $\tilde{\mathcal{Y}}_N$ are nondominated points.

Note that $\tilde{\mathcal{Y}}_N$ is not necessarily a subset of $\mathcal{Y}$ and there is no guarantee on quality of the points or how good of an approximation $\tilde{\mathcal{Y}}_N$ is. But still, having an approximation or *approximate nondominated front* is very important. The reason is although the goal of multi-objective optimization is to find the set of nondominated points, typically this is not easy (they are NP-hard to compute). Because, there may not exist an algorithm to find the entire front. Moreover, it may be very time-consuming to compute the exact nondominated front, as usually the number of nondominated points grows exponentially with the size of the problem (Boland, Charkhgard, and Savelsbergh 2016). There were lots of effort in finding the entire nondominated frontier such as the work of Boland, Charkhgard, and Savelsbergh 2015 on bi-objective mixed integer linear programs. For further information about exact methods on multi-objective integer linear programming, refer to the dissertation of Charkhgard 2016.

Although, it is impossible to exactly describe what a good approximation is in

terms of criteria such as closeness to the nondominated frontier, diversity, etc., it is possible to compare the quality of one approximate nondominated front with another (Zitzler et al. 2003). To do so, there are different approaches which one of them, namely hypervolume metric (Zitzler and Thiele 1999), will be discussed with more detail in Section III.D.

B. MOMP Methods Classification

According to Hwang and Masud 2012, the MOMP methods can be classified into three categories based on the involvement stage of a decision maker in the decision making process: The *a priori*, the *interactive*, and the *a posteriori* or generation methods.

In a priori methods the decision maker expresses his/her preferences in advance and before the start of the process. Methods such as *weighted sum*, where an importance weight is given to each criterion, is an example of a priori methods. One disadvantage of these methods is the difficulty of defining the weights or preferences in advance.

In the interactive methods, the decision maker is more involved throughout the decision making process. So, rounds of obtaining decision maker's preferences and calculations are done and in each step, the results are getting closer to the decision maker's most preferred solution. Besides requiring the decision maker to be more involved in the process (which is not always possible or even desirable by the decision maker), not knowing all the available options or an approximation of them are of drawbacks of the interactive methods.

In the a posteriori methods the efficient solutions of the problem or an approximation of them are presented to the decision maker and he/she can select the most preferred one among them. These methods provide the decision maker with all the information and nothing is left out (as opposed to the a priori and interactive

methods). The a posteriori methods are more computationally expensive.

The main focus in this research is on a posteriori methods. As mentioned in section III.A, acquiring nondominated front may not be possible and thus, approximation front is obtained. Therefore, heuristic and metaheuristic algorithms have been used to solve multi-objective optimization problems. Nondominated Sorting Genetic Algorithm (NSGA-II) of Deb et al. 2002, Pareto Archived Evolution Strategy (PAES) of Knowles and Corne 2000, Strength Pareto Evolutionary Algorithm (SPEA2) of Zitzler, Laumanns, and Thiele 2001, Multi-Objective Particle Swarm Optimization (MOPSO) of Coello, Pulido, and Lechuga 2004, Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) of Zhang and Li 2007, and Archived Multi-Objective Simulated Annealing (AMOSA) of Bandyopadhyay et al. 2008 are just to name a few of the well-known multi-objective evolutionary algorithms. Following a brief introduction to NSGA-II and PAES is discussed. For more comprehensive studies, refer to:

- The book of Deb 2001 on multi-objective optimization using evolutionary algorithms and the book of Coello, Lamont, and Van Veldhuizen 2007 on evolutionary algorithms for solving multi-objective problems.

- Jones, Mirrazavi, and Tamiz 2002 for a review of articles concerned with the theory and application of multi-objective metaheuristics from the period 1991–1999.

- Zhou et al. 2011 for a survey of multi-objective evolutionary algorithms up-to 2011.

- Marler and Arora 2004 for a survey of multi-objective optimization methods in the domain of engineering up-to 2004.

1. Multi-Objective Evolutionary Algorithms

The concepts and methodologies introduced in NSGA-II (Deb et al. 2002) and PAES (Knowles and Corne 2000) are used in this research. Therefore, an explanation of these algorithms is necessary.

a. Nondominated Sorting Genetic Algorithm, NSGA-II In each generation of NSGA-II, an offspring population Q_t of size N is created from a parent population P_t of size N by using usual genetic operators such as selection, crossover, and mutation (The first parent population P_0 is created randomly). The two populations are then combined to create R_t of size $2N$. Then *nondominated sorting* procedure applies on R_t.

The goal of nondominated sorting procedure is to rank and divide the individual solutions into different nondominated fronts. To achieve this, two entities are calculated for each solution: domination count n_p which represents the number of solutions which dominate the solution p and S_p as a set of solutions that the solution p dominates. Each solution p is compared with each solution q of the population. In a minimization problem, if $p \prec q \Rightarrow S_p = S_p \cup \{q\}$, and if $q \prec p \Rightarrow n_p = n_p + 1$. Solutions with $n_p = 0$ belong to the first front F_1. Now for each solution in the first front, each member q of its set S_p is visited and its domination count is reduced by one, namely $n_q = n_q - 1$. If n_q of any member becomes zero, they belong to the second front. By applying the same procedure on the second front, the third front is identified and so on. In the worst case, the $n_q = N - 1$ and there are at most $N - 1$ of such solutions. Having M objectives, the overall complexity of nondominated sorting procedure is $O(MN^2)$.

Next step after nondominated sorting of R_t, is to select a new parent population P_{t+1} of size N from R_t. So, selection begins from the solutions in the best front, namely F_1. If the size of F_1 is less than N, all F_1 members are added to P_{t+1}. This procedure continues by selecting the remaining members from subsequent fronts in

order of their ranking (F_2, F_3, and so on) until a complete front i cannot be included. In other words the number of solutions in fronts F_1 to F_i exceeds N and there is space only for a part of front F_i. In this case, the solutions from F_i are sorted using the *crowded distance* operator.

The goal of crowded distance operator is to increase diversity of solutions by assigning a higher rank to individual solutions that are from the least crowded regions. To compute crowding distance, first the solutions in the front are sorted according to each objective function value in ascending order. The boundary solutions with the smallest and largest values are assigned an infinite distance. For all the other solutions:

$$D_m{}^i = \frac{f_m{}^{i+1} - f_m{}^{i-1}}{f_m{}^{max} - f_m{}^{min}}$$

where $D_m{}^i$ is the distance of ith individual in the mth objective, $f_m{}^x$ is the value of the mth objective function of individual x, and $f_m{}^{max}$ and $f_m{}^{min}$ are the maximum and minimum values of the mth objective function. The overall crowding distance rank of a solution is the sum of the distance associated to each objective function, thus $D^i = \sum_m D_m{}^i$.

Figure. 6 shows the crowding distance concept in a bi-objective problem. The points with the same color are in the same nondominated front. The points indicated with 0 and l are the boundary points which are assigned an infinite distance to ensure they are always selected. To fill up the remaining space in P_{t+1}, the solutions are selected in descending order of their crowding distance rank.

The complete procedure of NSGA-II is depicted in Figure. 7. The overall runtime complexity of NSGA-II is $O(MN^2)$ which is governed by the nondominated sorting part.

b. Pareto Archived Evolution Strategy, PAES PAES has 3 forms: (1+1)-PAES, $(1+\lambda)$-PAES, and $(\mu+\lambda)$-PAES. The parameters in the parentheses indicate the number of parents and offsprings, respectively. So, for example in the $(\mu+\lambda)$-

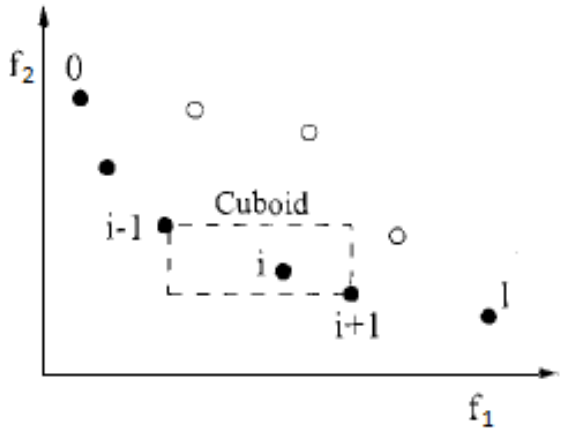


FIGURE 6 – Crowding distance calculation from Deb et al. 2002

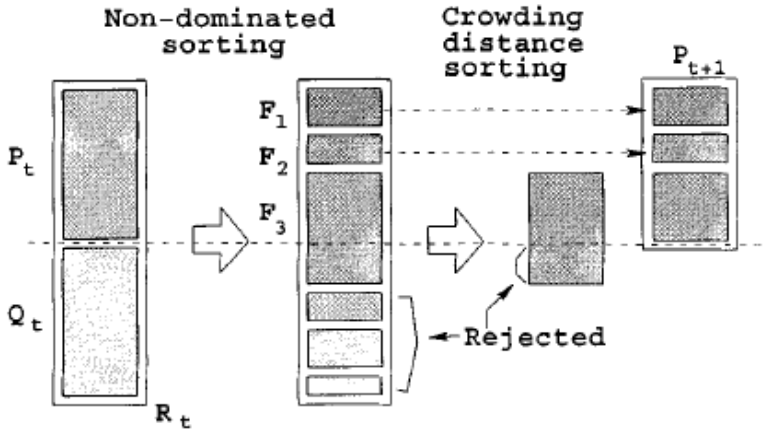


FIGURE 7 – NSGA-II procedure from Deb et al. 2002

PAES, λ offsprings are generated by mutating the μ parents. Here (1+1)-PAES is explained.

PAES has three main parts: generating the candidate solution, accepting or rejecting the candidate solution, and maintaining and *archive* of nondominated solution. First, an initial solution is generated randomly (parent). This solution is then mutated to create a new solution (offspring). Next, the offspring is compared to the parent. If the parent dominates offspring, the offspring is discarded and a new offspring is created. If the offspring dominates the parent, it is accepted and also replaced the parent to become the new parent for generating new solutions. But if none dominates the other, the offspring is compared with the nondominated solutions found so far that are kept in the archive.

If any element dominates offspring, offspring is discarded and a new candidate solution is generated. If offspring dominates any solution in the archive, it is accepted and all the dominated solutions are eliminated from the archive. If neither of the offspring or the archive dominate each other, then the density of the space where offspring can reside is checked against the parent. If offspring resides in a less crowded region, it is added to the archive and also accepted as the new parent.

In cases where the archive is reached its maximum capacity, the offspring region needs to be compared with the most crowded region of the objective space. If it does not reside in that region, then it is added to the archive and one of the members in the most crowded region is discarded.

The crowding region in the PAES algorithm is maintained by recursively dividing up the d-dimensional objective space where d is the number of objectives. Every time a solution is accepted, its grid location in the objective space is determined. This is done by repeatedly bisecting the range of each objective for finding the solution's grid. By setting l bisections of the space for each objective, the location of the solution is recorded as a binary string of length $2^{l \times d}$. If a solution's dimension

exceeds that of the archive, the ranges of the objective space are updated and the grid locations of all the solutions in the archive is recalculated. Along with the gird location of each solution, the number of solutions in each grid is also recorded to determine the crowded regions.

The worst case complexity of PAES for N evaluations is $O(aMN)$ where a is the archive length and M is the number of objectives. Since the archive size is usually chosen proportional to the population size N, the overall complexity of the algorithm is $O(MN^2)$. The whole process is described in the Algorithm 1.

Knowles and Corne 2000 reported that PAES, particularly in its baseline (1+1) form, is a capable multi-objective optimizer. In this research (1+1)-PAES is used as the main algorithm for solving the desired order picking problem. Before explaining how PAES is applied to the multi-objective mathematical model in II.B.1, first an illustrative example is provided and then, the solution structure is explained.

C. Solution Structure

1. Order Picking Example

To illustrate how any of the objectives in II.B.1 model come to play in picking an order, consider the following example. Assume two batches of the same item has occupied three locations. The stock of each batch at each location are shown in the Figure. 8. Assume, batch 1 expires 150 days later and batch 2 expires 100 days later. Also, the relative distance from I/O for the three different locations are 10, 20, 30 feet, respectively. It is assumed that the order size follows a normal distribution with mean of 11 and standard deviation of 2 units. Now suppose the next 5 orders for this item are {15, 7, 10, 11, 10}. What should be the best picking plan? How many and from which location(s) to pick to satisfy the orders while obtaining the best objective

Algorithm 1 Pseudo-code for (1+1)-PAES

Initialization: Create a random initial solution c and add it to the archive
current solution := initial solution
while stopping criteria not met **do**
    mutate c to create a new solution n and compare them
    if c dominates n **then**
        discard n
    else if n dominates c **then**
        replace c with n and add n to the archive
    else if n is dominated by any member of the archive **then**
        discard n
    else if n dominates any member of the archive **then**
        eliminate those members from the archive and add n to the archive
        apply TEST(c, n, archive) to determine the next current solution
    else
        apply TEST(c, n, archive)

function TEST(c, n, archive)
    if the archive is not full **then**
        add n to the archive
        if n is in a less crowded region of the archive than c **then**
            current solution = n
    else
        if n is in a less crowded region than some members of the archive **then**
            add n to the archive and remove a member of the archive from the most crowded region
            if n is in a less crowded region than c **then**
                current solution = n
        else if n is in a less crowded region than c **then**
            current solution = n
        else
            Discard m and c remains current solution

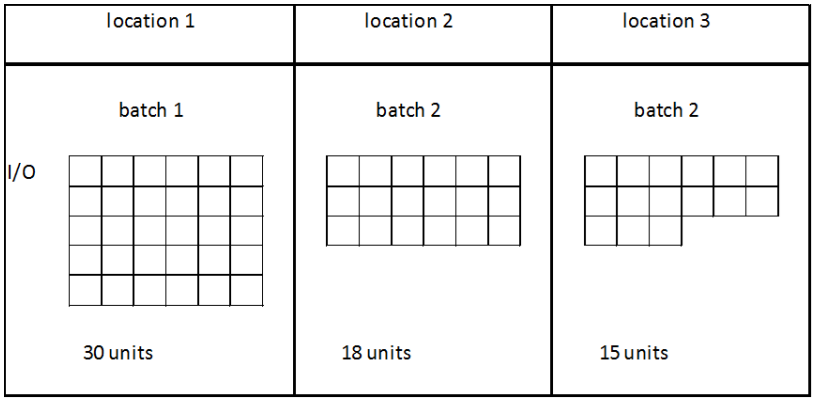


FIGURE 8 – Illustration of multiple batches and locations of an item

TABLE 1
ORDER PICKING EXAMPLE: SOLUTION 1

	Order 1			Order 2			Order 3			Order 4			Order 5		
Locations	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Picking Amount	15	0	0	0	0	7	0	10	0	11	0	0	0	8	2
Remaining	15	18	15	15	18	8	15	8	8	4	8	8	4	0	6

function values, i.e. the optimal solution? To answer these questions, three different solutions are investigated as shown in Tables 1–3.

In each table, the "Picking Amount" shows how each order is picked, i.e. how many and from which location. The "Remaining" row shows the stock remaining in each of the locations after the order pick. For instance in Table 1, one can see that to satisfy the first order (order of size 15), all 15 units are picked from location 1, nothing from location 2, and nothing from location 3. This changes the stock of the first location from 30 to 15 and the other locations are unchanged. Note that, since locations 2 and 3 are both from the same batch, one can split the order between these

TABLE 2
ORDER PICKING EXAMPLE: SOLUTION 2

	Order 1			Order 2			Order 3			Order 4			Order 5		
Locations	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Picking Amount	0	0	15	0	7	0	10	0	0	0	11	0	10	0	0
Remaining	30	18	0	30	11	0	20	11	0	20	0	0	10	0	0

TABLE 3
ORDER PICKING EXAMPLE: SOLUTION 3

	Order 1			Order 2			Order 3			Order 4			Order 5		
Locations	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Picking Amount	0	15	0	7	0	0	10	0	0	0	0	11	10	0	0
Remaining	30	3	15	23	3	15	13	3	15	13	3	4	3	3	4

two locations, a case that happened in picking the last order in solution 1.

So, which one of these 3 solutions should be chosen? To answer this question, the objective functions are evaluated. Tables 4-6, show the objective function values for each of the solutions ($f_1 - f_4$ in the mathematical formulation of II.B.1). Note that, f_3 in the model is comprised of two terms; one for the locations and one for the batches. These two terms are separated here and are shown by $f_3 - l$ and $f_3 - b$, respectively. These two terms should be maximized (or their negative be minimized) and all the other objective functions should be minimized. Also, f_1 relates to expiration date which just considers the batch rather than the location. So, locations 2 and 3 which are from the same batch, has the same value for expiration date function.

The last column in the above tables, show the summation of objective function value for the associated row. Comparing the above three solutions w.r.t the last column, it can be concluded that solution 2 dominates solution 1 (it has better objective values in all the functions) and solutions 2 and 3 are nondominated. At this stage, the decision maker can choose the solution based on his/her preferences.

It is worth noting that in this example, the warehouse can receive the 5 orders during 5 days, 5 weeks, one day, or any other time period. Time period is not important as long as the orders are picked separately. In this research if, for example, all the orders received throughout the day, are aggregated and picked at once, it is considered as one pick. In such situations, the lth order (in II.B.1 model) can be interchanged with the lth time period.

In this example, three arbitrary solutions out of many possibilities were inves-

TABLE 4: Order picking example: objective function values for solution 1

	Order 1			Order 2			Order 3			Order 4			Order 5			
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	**Total**
f_1	0.6		0	0		0.4	0		0.4	0.6		0	0		0.4	**2.4**
f_2	0.167	0	0	0	0	0.5	0	0.333	0	0.167	0	0	0	0.333	0.5	**2**
f_3_l	0.326	0.333	0.326	0.326	0.333	0.022	0.326	0.022	0.022	0.000	0.022	0.022	0.000	0.000	0.002	**2.083**
f_3_b	0.489		0.500	0.489		0.500	0.489		0.497	0.000		0.497	0.000		0.003	**3.463**
f_4	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	**14**

TABLE 5: Order picking example: objective function values for solution 2

	Order 1			Order 2			Order 3			Order 4			Order 5			
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	**Total**
f_1	0		0.4	0		0.4	0.6		0	0		0.4	0.6		0	**2.4**
f_2	0	0	0.5	0	0.33	0	0.17	0	0	0	0.33	0	0.17	0	0	**1.5**
f_3_l	0.333	0.333	0.000	0.5	0.25	0.000	0.5	0.25	0.000	0.5	0.000	0.000	0.309	0.000	0.000	**2.975**
f_3_b	0.500		0.500	0.500		0.500	0.250		0.500	0.250		0.500	0.309		0.000	**3.308**
f_4	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	**8**

TABLE 6: Order picking example: objective function values for solution 3

	Order 1			Order 2			Order 3			Order 4			Order 5			
	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	**Total**
f_1	0		0.4	0.6		0	0.6		0	0		0.4	0.6		0	**2.6**
f_2	0	0.333	0	0.167	0	0	0.167	0	0	0	0	0.5	0.167	0	0	**1.334**
f_3_l	0.333	0.000	0.326	0.333	0.000	0.326	0.280	0.000	0.326	0.280	0.000	0.000	0.000	0.000	0.000	**2.205**
f_3_b	0.500		0.500	0.500		0.500	0.421		0.500	0.421		0.011	0.000		0.011	**3.364**
f_4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	**15**

tigated. There can be solutions that dominate either solutions 2 or 3, be dominated by either one, or be nondominated solutions similar to them. Due to its small size, it is possible to find all the solutions for this problem and identify the nondominated set. However, as the size of the problem grows, it becomes harder and harder to do so. Therefore, finding the *approximate nondominated front* becomes the goal of solving this problem using PAES method. Unless noted otherwise, hereinafter, when PAES method is mentioned, its baseline (1+1) form is of interest.

2. Solution Representation in PAES

In PAES, the first step is to create an initial solution and then mutate it for generating an offspring solution. To create the initial solution, a schema similar to what was shown in Table 1 is used. So for each order, the eligible batches are identified. Eligible batches are those with enough stock to meet the received order. Next, a batch and a location are *randomly* chosen. If the location's stock is enough, all the order is picked from there. Otherwise, the remainder is picked from another location. Recall that order splitting between batches is not allowed but picking the order from multiple locations is allowed as long as they all hold the same batch. This process and evaluating all four criteria (objective functions) are repeated for each order.

There are two cases where the order of an item cannot be satisfied. First, if there is not enough inventory in any of the batches but the total inventory in all the locations is more than the order size. Second, if neither of the batches' inventory nor the total inventory of all the locations is sufficient. In the first case, there *could* be some solution(s) where the order is satisfied. So, all of the objective functions will be penalized. In the second case, however, there is not enough cumulative inventory. So, regardless of how and from where the previous orders have been picked, the new order could not be satisfied and thus, no penalty is necessary here. The complete

procedure for generating the initial solution is given in Algorithm 2.

Algorithm 2 Pseudo-code for generating initial solution in PAES

```
while an order left unprocessed do
    eligibility = IDENTIFY_BATCH(order)
    if eligibility = True then
        randomly choose a batch and a location
        if location has enough inventory then
            pick all the orders from there
        else
            randomly choose another location from that batch and pick the remainder from there
    else if eligibility = penalize then
        Penalize all the objective functions
    else
        This order could not be satisfied.

function IDENTIFY_BATCH(order)
    Find all the batches with enough stock to satisfy the order
    if eligible batch(es) are found then
        return True
    else if total inventory in all the locations > order then
        return penalize
    else
        return False
```

Having the initial solution, the next step is to mutate it for creating a new solution. *Offspring, child, candidate, new,* and *neighborhood* solutions are often used interchangeably. Several methods have been used to generate the neighborhood solution.

Single Point Mutation: An order is chosen randomly and the picking batch/location of that order is changed.

Double Point Mutation: Two orders are chosen randomly and their batch/locations are swapped, if different.

Double Point Swap: Two orders are chosen randomly and all the sequence of batch/locations for all the orders between these two are swapped.

Forward Single Point Mutation: An order is chosen randomly and the picking batch/location of that order and all the other orders following it are changed.

Random Strategy: An order is chosen randomly and out of 5 available strategies, one is chosen randomly. The 5 strategies are:

- Single point mutation.
- Pick from the closest location.
- Pick from the oldest batch (the closest expiration date). In case of multiple locations for the batch, select the location randomly.
- Pick from the location with the highest amount of inventory.
- Pick from the location with the lowest amount of inventory.

Forward Random Strategy: Similar to random strategy for the randomly chosen order and all the other orders following it.

In all these mutation methods, the batch-location eligibility is conserved, i.e. only batch-locations that have sufficient inventory are investigated. Besides the single point mutation, all the other methods in the "Random Strategy", aim to minimize one of the four criteria. The closest location, oldest batch, highest amount and lowest amount of inventory, aim at minimizing $f_1, f_2, f_3,$ and f_4, respectively.

In this problem, due to the dependency of each order pick to the preceding order, none of the above methods can be applied without a *repair procedure*. So, when the picking location of an order is changed for creating a new solution, the batch-location of all the following orders should be checked to ensure the feasibility of the solution. Because it is very probable to mutate the location of one order and make the succeeding order(s) infeasible.

For instance, consider solution 2 in Table 2 again. If order 4 is randomly chosen, the only eligible location for picking the 11 units is locations 1 (location 2 is

TABLE 7
ORDER PICKING PROFILE

Order	1	...	j	$j+1$	$j+2$	...	n
Order Size	O_1	...	O_j	O_{j+1}	O_{j+2}	...	O_n
Picking Location	L_1	...	L_j	L_{j+1}	L_{j+2}	...	L_n
Initial Inventory	I_1	...	I_j	I_{j+1}	I_{j+2}	...	I_n
Remaining Inventory	R_1	...	R_j	R_{j+1}	R_{j+2}	...	R_n

the current solution). By picking from location 1, its remaining inventory becomes 9. Without checking all the following orders and applying the necessary changes, the next order for 10 units which is originally picked from location 1, makes the solution infeasible. But by means of the repair procedure, the batch-location for order 5 is also changed (from location 1 to location 2).

For better illustration of repair procedure, consider order j in Table 7. If the picking location L_j changes, then R_j for the current and new locations will change as well. Remaining inventory of one order makes the initial inventory of the next order, i.e. $I_{j+1} = R_j$. Moreover, picking locations are chosen based on sufficiency of inventory in a location. So, if I_{j+1} changes, then L_{j+1} *might* change. This butterfly effect calls for an update of the initial inventories, remaining inventories, and picking locations of all the other orders as well.

So in generating the new neighborhood solution, if after any update of initial and remaining inventories of the orders, the picking location is still valid, it remains unchanged, otherwise, a new picking location should be chosen. It is worth noting that the idea of repair procedure has been appeared in other studies, such as Toledo et al. 2009 paper too. All of the mentioned solution generator methods, search the neighborhood randomly. The main reason, similar to the idea of the basic *variable neighborhood search* (VNS) method (Mladenović and Hansen 1997), is to avoid cycling which might occur in case of using deterministic rules. Also, randomized search of the neighborhood, increase the diversification of search in exploring nondominated solutions. To learn more about the principles and applications of VNS, refer to

Hansen and Mladenović 2001.

Knowing how the initial and neighborhood solutions are generated, the rest of the procedure is similar to the PAES algorithm explained in Algorithm 1.

Preliminary tests showed that the *forward single point mutation* and the *forward random strategy* performed better and generated better quality nondominated solutions than the others, with forward single point mutation performing slightly better. Therefore, it is chosen as the main neighborhood solution generator method. But how nondominated sets of different methods can be compared together? One way is to use the hypervolume metric, which is the subject of the next section.

D. Hypervolume

As explained in Section III.A, finding the Pareto optimal set is not always achievable in practice. Therefore, the aim of multi-objective optimization is to find an approximate nondominated front. But it is difficult to compare the approximation sets obtained from one algorithm with those of another algorithm. Because generally, one nondominated set is not dominantly better than the other. So there is a need for unary indicators or metrics to measure the quality of the solution sets. There are several indicators that try to describe the goodness of an approximation in terms of criteria such as closeness to the Pareto optimal set, diversity, and coverage of the space (Zitzler et al. 2003).

Hypervolume measure, also known as the S-metric (Zitzler 1999) or Lebesgue measure (Laumanns, Rudolph, and Schwefel 1999, and Fleischer 2003), is the best known indicator that was first introduced by Zitzler and Thiele 1999 as "size of the space covered". The hypervolume metric measures the amount of objective space that the approximation set dominates. In other words, it measures the volume of the approximate nondominated front w.r.t a reference point. Usually, the reference point is the worst possible point, i.e. nadir point. But the points with the worst objective

values in a dimension, cannot contribute in that dimension, as their distances from the nadir or reference point is zero. To avoid having a volume of zero, the worst point can be shifted (worsen) by an appropriate amount.

Comparing approximate nondominated frontiers, the one with a higher hypervolume is better. Because it is a better approximation of the true nondominated frontier. The hypervolume is maximized if and only if the set contains all Pareto optimal points (Fleischer 2003). Bringmann and Friedrich 2008 showed that the calculation of hypervolume metric is #P-hard (#P-hard of counting problems is analogous to NP-hard of decision problems), which implies that a polynomial-time algorithm does not exist, unless P = NP. Many researches have been done in this area to make the calculation faster. To learn more, refer to the dissertation of Bradstreet 2011.

In this research, the Hypervolume by Slicing Objectives (HSO) algorithm (While et al. 2006) is used for calculating the hypervolume indicator. Figure. 9 shows the hypervolume created by 4 points (a, b, c, d) in a three-objective maximization problem. This figure also shows how HSO sweeps along each objective and creates cross-sectional slices. In this figure, the unfilled circles represent points in each slice that are dominated by a point in y and z axes. Here, the reference point is the origin. The pseudo-code of HSO algorithm reproduced from While et al. 2006 and Bradstreet 2011 is shown in Algorithm 3.

The inputs of the algorithm are the nondominated front, f, and the number of objectives, n. Function "DEPTH" calculates the distance between two consecutive points. Function "INSERT" performs two tasks. First, it sweeps along the chosen objective and makes a *slice* between one point to the other. For doing so, it visits the points one at a time, skips the first objective of f, and add the points to a new list derived from $n-1$ objectives. So, it maintains a list of all the points that are visited and contributed to the current slice.

Second, it accumulates the list of points in each slice, ensures the points are

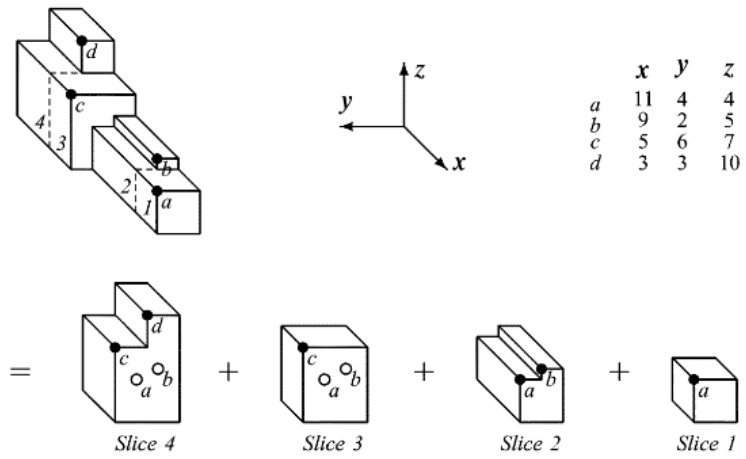


FIGURE 9 – HSO algorithm (While et al. 2006)

Algorithm 3 Pseudo-code of HSO

```
function HSO(f, n)
    f = sorted f in objective 1 in descending order
    if n = 2 then
        return the area created by the two dimensions of n
    np = empty list for keeping nondominated points
    for all p ∈ f do
        depth = DEPTH(f, p)
        INSERT(np, p, n − 1)
        hypervolume = hypervolume + depth × HSO(np, n − 1)
    return hypervolume

function DEPTH(f, p)
    if p is a one-dimension point then
        return distance from p to the reference point
    return distance from p to the next point in f w.r.t first dimension

function INSERT(np, p, n)
    skip the first objective and insert p to np
    remove any point in np that is dominated by p
    ensure all the points in np are sorted in the next relevant objective
```

nondominated and also sorted w.r.t to the next objective. So when a new points is added to the slice, all the points that are dominated by the new point are removed and the remaining points are sorted by their first objective. This concept is illustrated clearly in Figure. 9. So first, all the points are sorted in descending order of the first

objective (along x axis). Point a is the first to be visited and added to the list. Then, the slice from point a to point b is created (the first objective is skipped and the remainder is the 2-dimension area created by the values of y and z axes). Then point b is added and the slice now contains both a and b (they are nondominated w.r.t the remaining two objectives along y and z axes). When point c is visited, it dominates both a and b along y and z axes. So, they are removed and the slice contains only point c. Finally, point d is visited an added to the slice. Since each slice creates a new list of points with $n-1$ objectives, its hypervolume can be calculated recursively.

The HSO pseudo-code in Algorithm 3 is for maximization problem. The reference point in the simplest form can be the nadir point. However, in maximization problems (with all positive values), origin can be a good candidate which makes calculations easier. For minimization problem, on the other hand, there are two options. Either using the reference point, or normalizing the problem first, then inverting it and working with the normalized and inverted problem similar to a maximization problem. In this research, the latter approach has been used. Because it is easier to work with similar-signed values (all positive or negative) as opposed to different signs (which is the case in this research) and normalization takes care of this matter.

To normalize a point, a reference point is needed. Here, the reference point is the nadir point shifted (worsen) by an appropriate amount. So in a minimization problem, $f^{Ref} > f^N$, where f^N is the nadir point. Each point $f \in \tilde{\mathcal{Y}}_N$ is first normalized and inverted as follows:

$$f' = \frac{f - f^I}{f^{Ref} - f^I}$$
$$\bar{f} = 1 - f'$$

where $f^I, f', \text{and } \bar{f}$ are the ideal, normalized, and inverted points, respectively.

Note that, although a higher hypervolume is an indication of a better ap-

proximation, it is a point estimate. So, after having all the hypervolumes of all the solutions, a statistical hypothesis test (such as Kruskal-Wallis non-parametric test) is required for finding out which algorithm provided statistically significant better results.

Knowing the hypervolume indicator and the approximate nondominated front obtained using PAES method, one question is how they can help in generating the order picking strategies? The goal of next chapter is to answer this question.

CHAPTER IV
ORDER PICKING STRATEGIES

A. Why Rules?

In the previous chapter, the explanation for PAES multi-objective optimization algorithm, which is used in this research, was provided and the hypervolume indicator as a mean to measure the quality of approximate nondominated front (approximation) was introduced. Now assuming the MOMP model in II.B.1 is solved and the approximation for each item is obtained (one for each item). So, how a decision maker can decide which location should be visited for picking an order? If there are N items and the approximation of each item has M nondominated points (the archive size in the PAES method), then the decision maker should look at MN points and for each item, choose one point as the answer (N final points).

To avoid such rudimentary and time-consuming task, some patterns needs to be derived from the information in the sets of nondominated points of each item. These patterns can make the path for creating some order picking rules. One way is to classify items based on some common features. This results in a decrease in N. But still for all of the classes the M nondominated points need to be investigated.

Another option is to combine item classification with static rules. In other words, in generating the candidate solutions in multi-objective optimization algorithm, using static rules for directing the search, instead of randomly searching the neighborhood. To better understand this idea, consider the example in Figure. 8 again. This time, assume that the order size follows a normal distribution with mean

TABLE 8
ORDER PICKING EXAMPLE: STATIC RULE SOLUTION

	Order 1			Order 2			Order 3			Order 4			Order 5		
Locations	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3
Picking Amount	0	11	0	0	6	0	16	0	0	0	0	13	13	0	0
Remaining	30	7	15	30	1	15	14	1	15	14	1	2	2	1	2

of 11 and standard deviation of 5 units (as opposed to 2). Due to this standard deviation, one may consider this item a high variability item (there will be more discussion on these classifications later). Also, suppose the next 5 orders for the item are {11, 6, 16, 13, 13}.

For finding out the picking plan, the decision maker might take into account the high variability of the order size. Being concerned about the items expiration dates, he/she can suggest, for instance, to first pick from the older batch and if a batch exists in multiple location, the closest one be visited first. This rule is an example of a *static rule*. Although the picking location can change based on the stock availability in a location, but overall it gives a fixed suggestion. Based on this rule, the picking solution looks like Table 8.

Note that static rules decrease the number of solutions as they limit the number of choices. So, there will be fewer points in the decision space and thus, fewer points in the criterion space, namely smaller M. By combining item classification and static rules, there will be smaller MN combinations for the decision maker to consider. In the example above, the static rule created one solution. Depending on the rule, the number of solutions can be in the range of $(1, M)$ (In this setting, one may consider static rules as the more generic rules since random search of the neighborhood can also be a static rule and be treated as the *indifference* of the decision maker towards all the criteria, and thus, randomly selecting one for each order pick).

In this chapter, a few questions need to be addressed. How to classify the items? How to create a static rule (hereinafter, rule)? How to measure the quality of

the solution(s) generated by a rule?

B. Product Classification

One intuitive way to classify the products, is based on their order profiles, i.e. order size (volume) and order frequency, also known as popularity of product or number of times requested. These concepts play an important role in warehouse design and storage assignment. Decisions such as which SKUs and how many to store in the fast-moving area of the warehouse (if one exists), depends on the volume and frequency of a product. For example, an SKU with higher frequency, requires more visits. So, it adds more value to store this SKU closer or in the fast-moving area. To learn more about this, refer to the book of Bartholdi III and Hackman 2014.

Slotting decisions are not of a concern here. Slotting refers to the careful placement of each item within the warehouse (Bartholdi III and Hackman 2014) and it focuses on improving the efficiency of order picking. For example, an SKU that has an annual volume of 100 units and is picked in quantities of 5 units, has a frequency of 20 (the locations the SKU occupies, needs to be visited 20 times). Another SKU with the same volume which is picked in quantities of 50 units, is visited twice. From an order picking efficiency perspective, it is better that the first SKU is stored in more accessible locations.

Although slotting is not considered in this research, the characteristics of each SKU and each location in the warehouse are both considered. The criterion for each location is its *accessibility* which is a relative term. For example, in one warehouse this can be considered the physical position and a farther position is less favorable and less accessible than a closer one. In another warehouse, accessibility can be defined based on the required equipment. A closer location which needs a lift truck for higher shelves can be considered less accessible than a farther location at waist-height of the order picker.

There are more criteria attached to each SKU for consideration. Some of them are location-independent, such as the *order volume, variability,* and *frequency.* Others, depend on the locations each SKU occupies, such as *expiration date* and the *inventory* of the SKU in each location. Recall from Chapter I that the term *batch* was used for emphasizing on different expiration dates between SKUs of an item.

It is assumed, without loss of generality, that the accessibility of a location relates to its physical position. So, a closer location is considered more accessible and thus, terms such as *more accessible* and *closer* are used interchangeably. It is assumed that each batch is *randomly* stored in multiple locations. Due to the random storage, it is possible to investigate more general rules. Because one batch can be completely stored in closer or fast-moving locations and another one occupies both close and farther away locations.

For each batch, two levels of *low* and *high* are used for classifying order variability and frequency. So 4 different classes are made. The elements of these classes for the set of {order variability, order frequency} are:
{(*low, low*), (*low, high*), (*high, low*), (*high, high*)}.

For location-dependent criteria, three levels of expiration date, *early, middle,* and *late* and four levels of inventory, *exact, low, average,* and *high* are considered. Also, as explained above, a location is considered more accessible if it is closer. So, three levels of *close, middle range,* and *far* for the relative distance from the I/O is defined. So totally, there are $3 \times 4 \times 3 = 36$ different location-dependent classes. The elements of these classes for the set of {expiration date, distance, inventory} are:
{(*early, close, exact*), (*early, close, low*), ... (*late, far, high*)}.

The specifications of each class are explained more in the next section but to understand their purposes a little better, consider the following example. There is an item that belongs to the class of (*low, high*). So, it has a low order variability and high frequency of orders. Now this item, has two different batches (two different

expiration dates) and it is stored in three different locations. One of the locations is in the fast-moving area and the other two are farther away. The closer location still has some inventory in it and both of the farther away locations are completely full. Also, the closer location stores the batch which expires faster and the newer batch (which expires later) occupies both of the farther locations. So, the set of {expiration date, distance, inventory} for these three batches are:
{(*early, close, low*), (*late, far, high*), (*late, far, high*)}.

Now as explained in the previous section, these classifications help to decrease the number of points for investigation and they also help in defining the order picking rules. In the above example, the item is classified as low order variability and high order frequency. The decision maker does not need to worry about items expiring, because item's storage locations are visited frequently. Also, because of low variability, the inventory of a location will be consumed eventually (with a high probability). So, a closer location can be visited first. This represents another sample of a rule.

Note that all these classifications are for directing to a solution in the decision space. So, its image on all four objectives of model II.B.1 is, hopefully, in the approximate nondominated front. The specifications of each class and how to quantify them are discussed in the next section.

C. Class Specifications

The boundaries that separate the levels of the location-independent classes from one another, are relative to the warehouse and the products it holds. For example, in a warehouse that keeps fresh produce, the level of high frequency items should have a different threshold than a warehouse that stores garments. So, the boundaries used in this research are arbitrary and they can be easily modified depending on the products held in the warehouse. The boundaries for levels of order variability and frequency are explained in the next Chapter.

The order size is considered uncertain. So, to capture its variability, the coefficient of variation (CV) of the orders are used. For order frequency, the number of times an item is ordered annually is considered. There are two things here that need careful attention:

- CV is used to measure the variability of order size to ensure a high standard deviation does not necessarily result in a high variability (an item with mean of 6 and standard deviation of 2 shows more variability than an item with mean of 50 and standard deviation of 5).

- The variability indicator depends on the underlying distribution of the order size. For example, in a normal distribution, CV can be used as a measure of variation. But under Poisson distribution, CV is 1 which is not helpful (assuming the sample size was so small that it could not be approximated with a normal distribution).

Deciding on the levels of location-dependent classes are also relative to the warehouse and products, but it can be less restrictive in general. There are three levels for distance class. The warehouse is *uniformly* divided into three areas; close areas, far areas, and the ones in between (hence, middle range areas). So, one-third of the warehouse belongs to each of the three areas. To define the class level of a location, the term *relative distance* needs to be defined. Relative distance is the *ratio of a location's distance from the I/O to the maximum distance from the I/O* (farthest location). Any location with the relative distance of less than $1/3$ is in the *close* range. If the relative distance is more than $2/3$, the location is in the *far* range, and for any number in between the two, the location is in the *middle range*. Although dividing the warehouse into these three areas of accessibility is arbitrary, it provides more generic analysis. For example, an item with three different batches which can occupy three different locations, can be in any of the 27 different location combinations (There are

three areas of close, middle range, and far. Each batch can be in any of these areas: all in close, two in close and one middle, etc. Therefore, $3 \times 3 \times 3 = 27$ combinations).

In general, a popular strategy to increase the efficiency of order-picking processes is to divide the warehouse area into a fast-moving or forward area and a reserve or bulk area. The reserve area is used for replenishing the forward area and also, for picking items that do not exist in forward area. Deciding which items and in what quantities should be stored in the forward area, is not a trivial problem. Including an item in the forward area reduces the labor intensity of order-picking and implies replenishments from the reserve area should happen. Increasing the quantity of an item may reduce the number of replenishments but due to limited space of forward area, not all the items can make it there and benefit from the more accessible fast-moving area. In the literature, this problem is referred to as the forward-reserve problem (FRP). To learn more about FRP refer to initial works of Hackman, Rosenblatt, and Olin 1990, Frazelle et al. 1994, Berg et al. 1998, and also the book of Bartholdi III and Hackman 2014.

As mentioned before, slotting decisions are not of a concern and all the focus is on order picking rules in a more general and holistic settings. The goal of dividing the warehouse into three areas of close, middle range, and far, is to cover more warehouse layouts. In a warehouse with forward and reserve areas, some items are just in forward (close), some are just in reserve (middle and far), and some are in both. In a warehouse without a defined fast-moving area, every area can be considered bulk area. Even in this warehouse, some of those locations are closer to the shipping docks (close) and are easier or faster to access than other areas (middle and far). The classification in this research, works for these warehouses too.

By calculating the relative distance of a location, one can determine to which area does the item belong. For example consider two items, A and B. Both of them are stored in three different locations. The storage locations of item A are

80, 100, and 150 feet away from the I/O. For item B, the distances are 200, 700, 720 feet. The maximum distance in this warehouse from the I/O (the farthest location) is 1000 feet. So, the relative distance of the locations for item A are: $\{^{80}/_{1000}, ^{100}/_{1000}, ^{120}/_{1000}\} = \{8\%, 10\%, 12\%\}$ which are all less than $^{1}/_{3}$. So, the storage locations of item A are (*close, close, close*). The relative distance of the locations for item B are: $\{^{200}/_{1000}, ^{700}/_{1000}, ^{720}/_{1000}\} = \{20\%, 70\%, 72\%\}$. The first location is less than $^{1}/_{3}$ and the other two locations are greater than $^{2}/_{3}$. So, the storage locations of item B are (*close, far, far*).

There are also three levels for expiration date class which are *uniformly* distributed between *early, middle,* and *late* levels. Dividing the levels uniformly is arbitrary. Any other classification according to the warehouse and the importance of each expiration date level can be used instead. For any item, first the expiration date of each batch, e, is normalized w.r.t all the other batches of that item as follows:

$$e' = \frac{e - e^{\min}}{e^{\max} - e^{\min} + \epsilon}$$

where $e^{\min}, e^{\max},$ and e' are the minimum (earliest), maximum (latest), and normalized expiration date, respectively. In case the item has only one batch, denominator can become zero. This is why a very small number, ϵ, is added to the denominator to avoid such situation from happening. Then, each batch is put into the appropriate level based on its normalized expiration date. So, if $e' \leq {}^{1}/_{3}$, then the batch is in the *early* level. If $e' \geq {}^{2}/_{3}$, the batch is in the *late* level, and for any number in between the two, the batch is in the *middle* level.

For both expiration date and distance classes, no knowledge of the order size is necessary. But for inventory class, the order size should be known. It is assumed that the order size follows a normal distribution. Even if it does not, it is assumed there is enough historical data for order size of each item that its mean can be approximated

with a normal distribution (according to central limit theorem). To quantify the levels for inventory class, recall that the remaining inventory (hereinafter, remainder) R is the amount of inventory I after picking the order O, So $R = I - O$.

The available inventory of a location can be in any of the *exact, low, average,* and *high* levels. The *exact* level is when the inventory of a location is exactly the same as the order size. In this case, $R = 0$. *Low* refers to situations where the inventory is more than the order size and the demand can be satisfied but after the pick, the probability of the remainder satisfying a future order becomes very low. By considering a 95% confidence interval, a low level inventory can be written as:

$$P(R) \leq \Phi(-2) \Rightarrow R \leq \mu - 2\sigma$$

where μ and σ are the mean and standard deviation of the order size, respectively. Note that the 95% confidence interval in a standard normal distribution is the interval of $(\Phi(-1.96), \Phi(1.96))$. For simplicity, $(\Phi(-2), \Phi(2))$ is used instead. *Average* level is when the order can be satisfied from the inventory and after the pick, the remainder falls into the *low* level. So:

$$\Phi(-2) < P(R) \quad \& \quad P(R - \mu) \leq \Phi(-2) \Rightarrow \mu - 2\sigma < R \leq 2\mu - 2\sigma$$

High level is any other inventory level where the remainder becomes greater than or equal to the *average* level, which means:

$$P(R - 2\mu) > \Phi(-2) \Rightarrow R > 2\mu - 2\sigma$$

Note that locations with insufficient inventory are not considered as they cannot satisfy the order and thus, they cannot contribute in the solution. However, these locations still occupy the space. So, they should be considered in calculating the

space utilization (fourth objective function in model II.B.1).

The levels of each class should also be prioritized. For expiration date, the priority should be *early* $\succ$ *middle* $\succ$ *late* since in minimizing the objective function, it is preferred to first use the earlier expiry items (they have a lower numerical rank). With the same logic, levels of distance class are prioritized as *close* $\succ$ *middle range* $\succ$ *far*. The priority of inventory levels is *exact* $\succ$ *average* $\succ$ *high* $\succ$ *low*. Because in exact level, the order can be picked completely and the location is available. In other levels, the location is still in use but as going from average to high to low, the speed of the location becoming available decreases (Recall that with low, the chance of satisfying a future order are almost less than 2% but the location is still in use).

Up to this point, 36 location-dependent (hereinafter, *state*) and 4 location-independent classes (hereinafter, simply called *class*) are defined and their specifications are determined. Now consider an item in one of the 4 classes. Each location that this item occupies, can be in any of the 36 states. Potentially, each state is a solution. Because remember, the question to be answered is "from which location should be picked" and each state is an answer to this question.

There should be at least two batches of any item. Because with only one batch, the effect of expiration date will be canceled. For an item which occupies l locations, 36^l combinations of solutions is possible. In this research, 3 batches in 3 locations are considered, thus, $36^3 = 46656$ possible combinations of solutions. Of course not all these combinations are unique. For example, if all three batches are in the state of (*early, high, close*), then there is no difference in choosing one batch over the others. But if the three states are {(*early, close, low*), (*late, far, average*), ...(*early, far, high*)} then selection becomes difficult. Which of these three states should be preferred? The first batch expires soon and is also close, but picking from that location leaves a non-favorable remainder. The second batch expires later and is far but it has enough inventory for the current order and the remainder may satisfy

another future order. The batch in the third location expires soon and is far (so just by comparing these two properties with the first batch, it is not favorable). But it has high enough inventory to not worry about future orders. In this situation, none of the solutions dominate the other ones (they are all nondominating regarding one another).

It is important to find an approach to rank these solutions. That approach is the order picking *rule*. So a rule that says, "pick from the earliest expiry batch and in case of a tie, choose the one with a more reliable inventory" settles down the previous problem with three nondominating solutions. According to this rule, the batch in the third location is the preferred one.

Each solution is for one order pick and feeding it to the objective functions, results in a point. By applying a rule on a few consecutive orders of an item and comparing the final solution and its point with those of approximation points (obtained from applying PAES algorithm on the same problem), the quality of the rule can be shown.

The first step is to create these rules. Some rules can be very simple such as "picking from the closest location" or "picking based on first-expiry, first-out rule". The advantages of these practical rules is in their generality and simplicity. But they do not account for all the other criteria. As mentioned above, finding a rule is finding a way to prioritize the solutions. In other words, by finding the *relation* or the *pattern* between different states and the chosen solution(s), one can obtain rules in their rawest formats. Here, *association rule mining* is used for finding this relation.

D. Association Rule Mining

Association rule mining (ARM) is a method to find frequent patterns and relations between sets of items which was first introduced by Agrawal, Imieliński, and Swami 1993. ARM was initially used for market basket analysis to find the relations

between items purchased by customers. The data is in the form of transactions sets and each transaction contains sets of items. Consider I as the set of all items and two itemsets $X, Y \subset I$. An association rule is an implication of the form $X \Rightarrow Y$ where $X \cap Y = \emptyset$. Every rule is composed of two items sets; the left hand side or *antecedent* and the right hand side or *consequent*. In order to select significant rules among the set of all possible rules, a few concepts needs to be defined.

Definition IV.1. Support (sup): Fraction of transactions that contain both X and Y. $sup = P(X, Y)$

Definition IV.2. Confidence (conf): Measures how often items in Y appear in transactions that contain X. $conf = P(Y|X) = \frac{P(X,Y)}{P(X)}$

Definition IV.3. Lift: Computes the ratio between the rule's confidence and the support of the itemset in the rule consequent. $lift = \frac{sup(X,Y)}{sup(X)sup(Y)}$

In a nutshell, association rules are created by analyzing data for frequent *if-then* patterns and measures such as support, confidence, and lift help identifying the most important relationships. The goal of ARM is to find all the rules with $sup \geq minsup$ and $conf \geq minconf$ where *minsup* and *minconf* are the user-specified threshold for support and confidence, respectively.

Support indicates frequency of appearing the items in the database. A rule with a low support may occur simply by chance. Confidence indicates the number of times the relation between X and Y exists and it measures the reliability of the rule. So, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. Lift tries to rule out the random appearance of X and Y in a rule, given their individual support. So in a way, it indicates the strength of the rule. It is obvious from the definition of lift, that a value of 1 indicates independence of X and Y. Regardless of support and confidence, any lift < 1 provides no value (it has less strength even than the random occurrence of the itemsets).

To mine the association rules, the *Apriori algorithm* is used. Apriori is the best-known algorithm for mining association rules which was introduced by Agrawal and Srikant 1994. Briefly explaining, the algorithm tries to find all the *frequent itemsets* (itemsets with $sup \geq minsup$) iteratively, and then generate the rules using these frequent itemsets.

In this research, simple rules are of interest. Simple rule is a rule with a single item as its consequent. To mine the rules, first the transactions need to be created. Each combination of the states is a partial transaction (partial because it is not complete and something else should be added). For example, three batches in three states of $\{S_1 : (\textit{early, close, low}), S_2 : (\textit{late, far, average}), S_3 : (\textit{late, far, high})\}$ comprise the partial transaction. Now, this transaction will be complete if the priority of these states are known. In this example, the S_1 and S_2 are nondominated and $S_3 \preceq S_2$. So, it can be removed from consideration (pairwise comparison between each elements of states is done. State 2 weakly dominates state 3 and is nondominated w.r.t state 1). So, S_1 and S_2 can be added to the transaction list to complete it.

Here the transactions are comprised of the three states of the three batches and the nondominated states (after the pairwise comparisons). To distinguish between them in the transaction T, they are shown as follows:
$T = \{(\textit{early, close, low}), (\textit{late, far, average}), (\textit{late, far, high}), S_1, S_2\}$.
After generating all the transactions (all the combinations of states and their pairwise comparisons), the rules can be mined by using the Apriori algorithm. The "arules" package in R (Hahsler, Gruen, and Hornik 2005) is used to achieve this goal. Table 9 shows the refined results of mining simple rules based on all the states for three batches in three locations.

In this table the "order" column, is the rank of each rule and in cases of ties, the highest rank of that group is given to all (the lower the number, the higher the rank). The first row of the table (first rule) says, if there are three batches and at

TABLE 9
SIMPLE RULES WITH ORDERS BASED ON CONFIDENCE

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Support**	**Order**
Early	Close	Exact	1	0.081	1
Middle	Close	Exact	0.944	0.076	2
Early	Middle Range	Exact	0.944	0.076	2
Early	Close	Average	0.944	0.076	2
Late	Close	Exact	0.890	0.072	5
Early	Far	Exact	0.890	0.072	5
Early	Close	High	0.890	0.072	5
Middle	Middle Range	Exact	0.838	0.068	8
Middle	Close	Average	0.8388	0.068	8
Early	Middle Range	Average	0.839	0.068	8
Early	Close	Low	0.839	0.068	8
Late	Middle Range	Exact	0.738	0.060	12
Middle	Far	Exact	0.738	0.060	12
Late	Close	Average	0.738	0.060	12
Early	Far	Average	0.738	0.060	12
Middle	Close	High	0.738	0.060	12
Early	Middle Range	High	0.738	0.060	12
Middle	Middle Range	Average	0.644	0.052	18
Middle	Close	Low	0.644	0.052	18
Early	Middle Range	Low	0.644	0.052	18
Late	Far	Exact	0.6	0.049	21
Late	Close	High	0.6	0.049	21
Early	Far	High	0.6	0.049	21
Late	Middle Range	Average	0.476	0.039	24
Middle	Far	Average	0.476	0.039	24
Middle	Middle Range	High	0.476	0.039	24
Late	Close	Low	0.476	0.039	24
Early	Far	Low	0.476	0.039	24
Middle	Middle Range	Low	0.333	0.027	29
Late	Far	Average	0.271	0.022	30
Late	Middle Range	High	0.271	0.022	30
Middle	Far	High	0.271	0.022	30
Late	Middle Range	Low	0.124	0.010	33
Middle	Far	Low	0.124	0.010	33
Late	Far	High	0.071	0.006	35

least one of those batches is in the state of (*early, close, exact*), the pick should happen from that batch. The confidence of this choice is 1 or 100%. So technically, this is the ultimate winner. This one is not surprising at all. Because a batch in that state, minimizes all the objective functions. It has the best expiration date (early), closest location, and the inventory of the location exactly matches the amount of order. Following the same logic, the state (*late, far, low*) has the worst values for all the objectives and it will always be the loser state. This is why it does not exist in any of the rules.

As one goes down the list, the confidence becomes smaller. So for example, if there are three batches and one of them is in the state of (*late, close, high*), this is not a clear winner but still, in 60% of the times, this batch was one of the winners (one of the nondominated states). Recall that confidence is an indicator of the reliability of the rule and how many time the relation exists (relation between existence of this state as one of the three states and then, this state making it to the list of nondominated states).

There are many more simple rules than the 35 shown in Table 9. But some of them do not provide a great deal of insight. For instance, consider the following rule which does not provide enough information:
"Regardless of the expiration date class, if one batch is in the state of (*far, high*), then with $X\%$ confidence a batch is in the state of (*close, low*) exists."
Although, support of this rule indicates the frequency of these two states appearing together in all of the combinations of the states, but more information can be derived from knowing about the state of each batch individually (this is an exclusive case for the problem in this research and by no means is inclusive to all the situations that ARM is utilized).

An astute reader may find some of the rules counter-intuitive. Why does the state (*early, close, low*) have the same confidence as (*early, middle range, average*) and

has a higher confidence than (*middle, far, exact*)? The reason for this is very simple. All these rules are class-agnostic, as if a completely *indifferent decision maker* ranked them. So if a rule weakly dominates another rule, it surely has a higher confidence. But if not, the confidence goes to each individual element of the state. The first two classes of the state (*early, close, low*) are at their best levels and the last one at its worst. However, (*middle, far, exact*) has one class at its best, one at the worst and one in between. So, although these two states are both considered nondominated w.r.t one another, but the first has clearly more favorable states from a rule mining perspective.

As mentioned before, these rules are in their rawest format. So to account for various preferences of the objective functions, states, and differences between classes of items, these rules need to be modified. The rules in Table 9 are used as the baseline scenario. Each modified rule can then generate a point for the multi-objective optimization problem. This point, can be compared to any other points created by other rules and also to the approximate nondominated front for an evaluation of its quality. Rules which could create an acceptable gap from the points of the approximate nondominated front will become the desired order picking strategies.

There are three outcomes out of comparing the rule's point with those of the approximation. The rule's point is worse (dominated by at least one point of the approximation), it is not dominated by the approximation so it becomes one point of the front, or it dominates some or all of the points in the approximation front. Note that the latter might happen because there is no knowledge of the true nondominated front and only the approximation is available through the use of multi-objective optimization.

Tables 10 and show some customized rules. Note that, the confidence and support are the same as those seen in Table 9 and just the orders are changed to reflect the priorities in selecting one state over another. So, only confidence is shown

in the tables for a reference. Table 10 shows some customized rules where the order of priority is expiration date, distance, and inventory level, respectively. Here, the natural ordering of each class is considered, i.e. the smaller the value, the higher the rank. So, unlike the dominance relation explained before, a *low* inventory level is ranked higher than *average* and *high* levels. Note here there is no tie in the order column as the confidence of each row is different from its immediate follower.

Table 11 is more customized and considers the interaction between the classes. Here, first the priority is given to all the states with the *exact* inventory level and these states are ordered based on the expiration date. After reaching the *average* inventory level, the *early* expiration date is given higher priority. Also, all the *low* inventory levels are put at the end. Because picking up from *low* inventory locations will leave an unpleasant remainder. But since the future orders are uncertain, they may become an *exact* location. Similar to the *exact* level, the *low* inventory level states are ordered based on the expiration date. The other states can be ordered according to the previous states. For example, (*late, close, ave*) is put ahead of (*middle, close, high*). Because as time passes, the first state can become (*middle, close, average*) with the order of 13 and the second can become (*early, close, high*) with the order of 14.

Different item classes can have an effect on the rules as well. Order frequency of an item has more effect on the decisions regarding the distance and expiration date. If the order frequency of an item is high, its storage locations are visited more often. In this case, picking from the closer and faster picking areas can be more favorable than considering how fast a location becomes empty (inventory objective) or whether the earliest expiry batch is being selected (expiration date objective). On the other hand, when the order frequency is low, the item may expire before the next order comes. So, a batch with earlier expiration date will be in the focus of attention.

Order variability of an item has more effect on the decisions regarding in-

TABLE 10
SIMPLE RULES WITH PRIORITIZED ORDERING OF EXPIRATION DATE, DISTANCE, AND INVENTORY, RESPECTIVELY

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Early	Close	Low	0.838	2
Early	Close	Average	0.944	3
Early	Close	High	0.89	4
Early	Middle Range	Exact	0.944	5
Early	Middle Range	Low	0.644	6
Early	Middle Range	Average	0.838	7
Early	Middle Range	High	0.738	8
Early	Far	Exact	0.89	9
Early	Far	Low	0.476	10
Early	Far	Average	0.738	11
Early	Far	High	0.6	12
Middle	Close	Exact	0.944	13
Middle	Close	Low	0.644	14
Middle	Close	Average	0.838	15
Middle	Close	High	0.738	16
Middle	Middle Range	Exact	0.838	17
Middle	Middle Range	Low	0.333	18
Middle	Middle Range	Average	0.644	19
Middle	Middle Range	High	0.476	20
Middle	Far	Exact	0.738	21
Middle	Far	Low	0.124	22
Middle	Far	Average	0.476	23
Middle	Far	High	0.271	24
Late	Close	Exact	0.89	25
Late	Close	Average	0.738	26
Late	Close	High	0.6	27
Late	Middle Range	Exact	0.738	28
Late	Middle Range	Average	0.476	29
Late	Middle Range	High	0.271	30
Late	Far	Exact	0.6	31
Late	Far	Average	0.271	32
Late	Far	High	0.071	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36

TABLE 11
SIMPLE RULES WITH CUSTOMIZED ORDERING

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Early	Middle Range	Exact	0.944	2
Early	Far	Exact	0.89	3
Middle	Close	Exact	0.944	4
Middle	Middle Range	Exact	0.838	5
Middle	Far	Exact	0.738	6
Late	Close	Exact	0.89	7
Late	Middle Range	Exact	0.738	8
Late	Far	Exact	0.6	9
Early	Close	Average	0.944	10
Early	Middle Range	Average	0.838	11
Early	Far	Average	0.738	12
Middle	Close	Average	0.838	13
Early	Close	High	0.89	14
Early	Middle Range	High	0.738	15
Middle	Middle Range	Average	0.644	16
Middle	Far	Average	0.476	17
Late	Close	Average	0.738	18
Middle	Close	High	0.738	18
Early	Far	High	0.6	20
Middle	Middle Range	High	0.476	21
Late	Middle Range	Average	0.476	21
Late	Close	High	0.6	23
Late	Far	Average	0.271	24
Middle	Far	High	0.271	24
Late	Middle Range	High	0.271	24
Late	Far	High	0.071	27
Early	Close	Low	0.838	28
Early	Middle Range	Low	0.644	29
Early	Far	Low	0.476	30
Middle	Close	Low	0.644	31
Middle	Middle Range	Low	0.333	32
Middle	Far	Low	0.124	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36

ventory level. A high variability order brings more concern to the locations where the remaining inventory level for the future picks are favorable. For example, (*far, far, exact*) becomes more favorable than (*early, close, low*) regardless of the higher confidence of the latter in the basic rules. The interaction of order frequency and variability should also be considered in the rules. Tables 12–14 show some rules considering the effect of different item classes. Table 11 can be considered as one for low frequency and high variability.

Aside from the rules in Table 9 which are ordered based on confidence, one can find other rules only based on the absolute priority of one class over the others. For example, assume the order of priority is expiration date, distance, and inventory. In this case, if one batch's expiration date is earlier than the other ones, then regardless of its location and inventory level, the pick should be done from this batch. Following the same logic, there are $3! = 6$ rules solely based on the priority of one class over the others. Although in general this may not be the case, these rules can serve as baseline scenarios for more customized rules.

1. Evaluation of Rule Quality

Each of the introduced rules can create one point. To evaluate the quality of a rule against another rule, one can check their dominance relations against each other. But often time this is not sufficient as the rules can be both non-dominating or in some cases, rules create more than one point. In these cases, it is possible to measure the quality of each rule by calculating their hypervolume. The one with a larger hypervolume, covers more of the decision space and therefore, generates better solution.

In evaluating the quality of a rule's point against that of approximated non-dominated front of PAES algorithm, some adjustments should be taken into account to make the comparison fair. Because here, the quality of one point is judged against

TABLE 12
SIMPLE RULES WITH CUSTOMIZED ORDERING FOR HIGH FREQUENCY AND LOW VARIABILITY

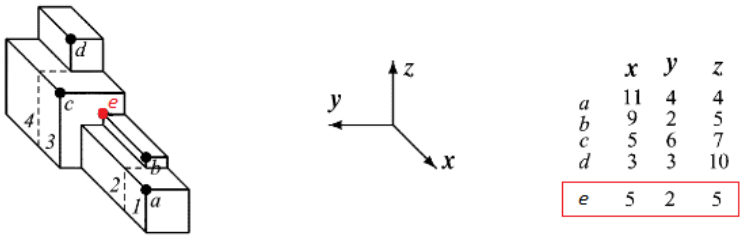
Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Middle	Close	Exact	0.944	2
Late	Close	Exact	0.89	3
Early	Middle Range	Exact	0.944	4
Middle	Middle Range	Exact	0.838	5
Late	Middle Range	Exact	0.738	6
Early	Far	Exact	0.89	7
Middle	Far	Exact	0.738	8
Late	Far	Exact	0.6	9
Early	Close	Average	0.944	10
Early	Close	High	0.89	11
Middle	Close	Average	0.838	12
Late	Close	Average	0.738	13
Middle	Close	High	0.738	13
Late	Close	High	0.6	15
Early	Middle Range	Average	0.838	16
Early	Middle Range	High	0.738	17
Middle	Middle Range	Average	0.644	18
Middle	Middle Range	High	0.476	19
Late	Middle Range	Average	0.476	19
Late	Middle Range	High	0.271	21
Early	Far	Average	0.738	22
Middle	Far	Average	0.476	23
Late	Far	Average	0.271	24
Early	Far	High	0.6	25
Middle	Far	High	0.271	26
Late	Far	High	0.071	27
Early	Close	Low	0.838	28
Early	Middle Range	Low	0.644	29
Early	Far	Low	0.476	30
Middle	Close	Low	0.644	31
Middle	Middle Range	Low	0.333	32
Middle	Far	Low	0.124	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36

TABLE 13
SIMPLE RULES WITH CUSTOMIZED ORDERING FOR LOW FREQUENCY AND LOW VARIABILITY

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Early	Middle Range	Exact	0.944	2
Early	Far	Exact	0.89	3
Early	Close	Average	0.944	4
Early	Middle Range	Average	0.838	5
Early	Far	Average	0.738	6
Early	Close	High	0.89	7
Early	Middle Range	High	0.738	8
Early	Far	High	0.6	9
Middle	Close	Exact	0.944	10
Middle	Middle Range	Exact	0.838	11
Middle	Far	Exact	0.738	12
Late	Close	Exact	0.89	13
Late	Middle Range	Exact	0.738	14
Late	Far	Exact	0.6	15
Middle	Close	Average	0.838	16
Middle	Middle Range	Average	0.644	17
Middle	Far	Average	0.476	18
Middle	Close	High	0.738	19
Middle	Middle Range	High	0.476	20
Middle	Far	High	0.271	21
Late	Close	Average	0.738	22
Late	Middle Range	Average	0.476	23
Late	Far	Average	0.271	24
Late	Close	High	0.6	25
Late	Middle Range	High	0.271	26
Late	Far	High	0.071	27
Early	Close	Low	0.838	28
Early	Middle Range	Low	0.644	29
Early	Far	Low	0.476	30
Middle	Close	Low	0.644	31
Middle	Middle Range	Low	0.333	32
Middle	Far	Low	0.124	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36

TABLE 14
SIMPLE RULES WITH CUSTOMIZED ORDERING FOR HIGH FREQUENCY AND HIGH VARIABILITY

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Middle	Close	Exact	0.944	2
Late	Close	Exact	0.89	3
Early	Middle Range	Exact	0.944	4
Middle	Middle Range	Exact	0.838	5
Late	Middle Range	Exact	0.738	6
Early	Far	Exact	0.89	7
Middle	Far	Exact	0.738	8
Late	Far	Exact	0.6	9
Early	Close	Average	0.944	10
Middle	Close	Average	0.838	11
Late	Close	Average	0.738	12
Early	Close	High	0.89	13
Middle	Close	High	0.738	14
Late	Close	High	0.6	15
Early	Middle Range	Average	0.838	16
Middle	Middle Range	Average	0.644	17
Late	Middle Range	Average	0.476	18
Early	Middle Range	High	0.738	19
Middle	Middle Range	High	0.476	20
Late	Middle Range	High	0.271	21
Early	Far	Average	0.738	22
Middle	Far	Average	0.476	23
Late	Far	Average	0.271	24
Early	Far	High	0.6	25
Middle	Far	High	0.271	26
Late	Far	High	0.071	27
Early	Close	Low	0.838	28
Early	Middle Range	Low	0.644	29
Early	Far	Low	0.476	30
Middle	Close	Low	0.644	31
Middle	Middle Range	Low	0.333	32
Middle	Far	Low	0.124	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36



	x	y	z
a	11	4	4
b	9	2	5
c	5	6	7
d	3	3	10
e	5	2	5

FIGURE 10 – Grand hypervolume

the quality of multiple points. The idea for a fair comparison, is to first combine all the nondominated points of both PAES algorithm and the rule to makes up the *grand approximation*. This can be consisted of only PAES points (if at least one of PAES points dominates the rule's point), only rule's point (if it dominates all the points of PAES), or a combination of both (if rule's point does not dominate all the PAES points or when neither of them is dominating the other).

For a better illustration of this concept, consider a modification of Figure. 9 in the following example. Assume points $a - d$ in Figure. 10 are from the PAES algorithm and point e is from a rule. Clearly, the grand approximation is all the points of PAES as points b and c dominate point e. Thus, grand hypervolume is the same as PAES hypervolume (h_1). So, the percentage contribution of h_1 to the grand hypervolume is 100%.

Now for a fair comparison, the hypervolume generated by point e is calculated under the absence of dominating points. In other words, considering all the approximate nondominated points, how much point e could contribute to the hypervolume of the space if it was part of the approximate nondominated front and non of the points dominating it, were existed. Note that the grand approximation space is not smaller than any of the PAES or the rule's decision space. Therefore, in a maximization problem,

$$f^N{}_{grand} \preceq f^N{}_{PAES} \text{ and } f^N{}_{grand} \preceq f^N{}_{rule}$$

and

$$f^I{}_{grand} \succeq f^I{}_{PAES} \text{ and } f^I{}_{grand} \succeq f^I{}_{rule}$$

where $f^N{}_x$ and $f^I{}_x$ are the nadir and ideal point of space x, respectively.

Recalling from the explanation of NSGA-II algorithm in Section III.B.1.a, this means bringing the rule's point to the first front (F_1) by removing all the points in F_1 that dominates it. In cases that the rule's point dominates at least one point of the PAES method, it is already in F_1. So the procedure for the overall calculation is as follows:

1. Combine PAES points and the rule point and create the grand approximate nondominated front (grand approximation).

2. Calculate the grand hypervolume (h_g) using the grand approximation.

3. Calculate the percentage contribution of PAES points and the rule point to the grand hypervolume.

    - Calculate PAES hypervolume (h_1) in the space of grand approximation.
    - Remove any point from the grand approximation that dominates rule's point and calculate the hypervolume of all the remaining points (h_2).
    - Calculate the percentage contribution of PAES points by $\frac{h_1}{h_g}$ and the rule's point by $\frac{h_2}{h_g}$.

4. The percentage gap can be calculated as the difference between the percentage contributions, i.e. $gap = \frac{h_1 - h_2}{h_g}$.

If *gap* is positive, PAES points contribute more to the decision space, and if negative, the rule's point could contribute more. All of this, is calculating the gap between the points of PAES algorithm with that of a specific rule. So, this process can be simulated multiple times for different problems and the gap is calculated each time.

Average of the gaps can show the average percentage gap of the rule from the PAES frontier and serve as an indicator of a given rule's solution quality.

2. Generating Points From The Solutions

It was mentioned earlier that each rule can generate a point for the multi-objective optimization problem. So a solution like "picking from the batch in (*early, close, high*) state", should provide a point such that it is comparable with the values generated by PAES algorithm. The first step in generating points from the solutions is quantifying the states. This part is already taken care of in definition of each class. For example, as explained in Section IV.C, any location with the relative distance of less than 1/3 is in the *close* range. If the relative distance is more than 2/3, the location is in the *far* range, and for any number in between the two, the location is in the *middle range*.

The second step is to find the link between the three elements of each state (distance, expiration date, inventory) and the four desired criteria (distance, expiration date, space usage, remainder). For the distance objective function, the distance of the chosen batch from I/O is considered. Space usage objective function can be derived from the locations' stocks. If there is any inventory in a location, it is in-use and thus, it is considered in the calculations. Space usage is for all the batches and not just the one selected for the pick. So, the number of all the available locations is used for the space usage objective function.

For every pick, the expiration dates of all the batches are used in calculating the expiration date objective function (or the negative of salvage function for that matter). When one batch is completely depleted, it affects the state of other batches as well. So if there are three batches with (*early, early, late*) expiration dates, and the *late* one consumed faster, it is necessary to calculate the state of the other two batches again to see relatively which one is *early* and which is *late*. So, for picking

decisions, the states should be updated after each pick. But in evaluating the picking decision, the order of the batches should be kept unchanged. In other words, the initial states of the batches should be kept on record for evaluating the decision. The following example clarifies this concept better.

Assume there are three batches in the states of $\{S_1 : (\textit{early, close, high}), S_2 : (\textit{early, middle range, high}), S_3 : (\textit{late, far, average})\}$ and they are related to an item with low frequency and high order variability. The inventory of these batches are 15, 35, and 10, respectively. For the sake of this example, the picks from the location are scored 0, 1, 2, if the pick is from a batch with (*early, middle, late*) expiration date level respectively. These scores are used for the expiration date objective function. Now assume an order of size 10 is received and the third batch is selected for the pick. The other two batches should be updated after this pick. The new update brings the batches to the states of $\{S_1 : (\textit{late, close, high}), S_2 : (\textit{early, middle, high})\}$. Two new orders of sizes 5 and 10 are received and the first batch is chosen for both picks. Now assume another picking plan where the first and second order are picked from the first batch and third order is picked from the third batch.

The expiration date level of the batches are relative to the other batches and after each pick they should be updated. However, in the second plan, the updates after the picks would not change the levels for any of the batches. For scoring purposes, however, the initial level of the batches should be used. Otherwise, in the first plan, the total picking score is $2+2+2 = 6$ and in the second plan the score is $0+0+2 = 2$. The second plan has a better score while essentially both picks are targeting the same two batches (in the minimization problem, the smaller the score, the better). To avoid such mistakes from happening, the initial level of each batch should be used. In that case, both plans have a score of 2. So in summary:

The expiration date levels of the batches are updated for picking decisions, but the initial levels of the batches are kept on record for objective function

calculation purposes.

For the item's remainder objective function, the inventory at each location after the pick (hence, remainder) is considered. Having four inventory levels before the pick, namely *exact, low, average,* and *high*, five remainder inventory at each location after the pick can be obtained. *Zero, insufficient, low, average,* and *high* where both *average,* and *high* can be obtained from the *high* level. Since considering available inventory for the future is of interest, the *zero* level can be dropped from the calculation.

Borrowing from the definition of each inventory level, the score of the remainders should reflect the average number of orders they can satisfy in the future. So, for *insufficient, low, average,* and *high* remainders, the score will be $0, 1, 2,$ and $3+$. Mathematically speaking:

$$score = \left\lfloor \frac{R + 2\sigma}{\mu} \right\rfloor$$

This relation comes from the definition of each inventory level. For example *low* is an inventory level where current order can be satisfied but the probabilities of the remainder satisfying a future order is very low. So, at least one order but not necessarily two orders can be picked from a *low* inventory location. From its definition and with basic algebra:

$$R \leq \mu - 2\sigma \Rightarrow \frac{R + 2\sigma}{\mu} \leq 1$$

By applying the same concept to all the other levels, the RHS of the above relation gives the average number of future orders that can be satisfied. Since the inventory of an item is an integer value, the floor of $\frac{R + 2\sigma}{\mu}$ is used. This score is calculated for all the available locations (i.e. locations with inventory) and its average is used as the objective function value of item's remainder. The higher the score (the lower the negative score) the better.

In this chapter, the reasons for introducing the rules were discussed, rules were

mined using Apriori algorithm, they were modified to reflect different item classes and preferences, they were quantified and linked to the objective functions, and the ways to evaluate their quality against PAES algorithm was explained. The next chapter will walk through the steps of applying these rules to sample data and discuss the results.

CHAPTER V
COMPUTATIONAL STUDIES

In this chapter, both PAES algorithm for solving the multi-objective order picking optimization problem (discussed in model II.B.1) and the rules generated in the previous chapter are put into practice. First, they are applied to randomly generated data to see what insights can be concluded from their results. Then, their performances on some real-world problems are evaluated. The algorithms in this chapter were coded in Visual C# and all computational runs were from a 2.50 GHz Intel Core i7 with 16 GB of RAM.

A. Randomly Generated Data

1. Data Structure

In all of the randomly generated instances, the following parameters dictate the size of the problem.

$v :=$ order size variability $= 2$

$f :=$ order frequency $= 2$

$m :=$ number of products in each class $= 30$

$n :=$ number of batches of each product $= 3$

$p :=$ number of locations $= 90$

$t :=$ number of orders $= 30$

Two different classes of *low* and *high* are considered for both v and f. Since the order sizes are following a normal distribution, all the data with a CV in the range of $(0.05, 0.2)$ are classified as *low variability* and those with CV in the range of $(0.3, 0.5)$ are classified in the *high variability* group. The order frequency was considered as the number of times an item is ordered annually. Converting that to the daily operations, if an item has an order frequency of more than one, it is classified as the *high frequency* item, otherwise, it is classified in the *low frequency* group. 30 different products in each class are introduced and it is also assumed without loss of generality that $p = m \times n$. So, three batches of each product are stored in three different locations. Here, t is the number of times order picking is required and has nothing to do with the size of the orders. So for each item, 30 trip is necessary.

The parameters for each individual product in any class are created by randomly assigning:

- The distribution parameters of the order size (μ and CV of the order size which follows a normal distribution);
- Storage locations;
- Expiration dates of each batch;
- Initial inventory in each location;
- Order volume for each individual pick.

The ranges of CV was mentioned above. For μ random numbers were generated from two arbitrary ranges; namely $(2, 10)$ or the range of $(20, 40)$. Different expiration dates were assigned randomly to each batch of any product to ensure they are different and then, they were randomly assigned to different storage locations. For initial inventory

of each item's batch in each location, the following formula was used:

$$s_{ik} = \left\lceil \frac{(\mu_i + \beta\sigma_i)t}{n} \right\rceil$$

where s_{ik} is the inventory of item i in the kth location, μ_i and σ_i are the mean and standard deviation of the order size for item i and β is a random number between $(1, 4)$.

The lth order volume of each item i was generated using the following relation:

$$o_{il} = \lfloor \mu_i + |Z_\alpha|\sigma_i \rfloor$$

where α is a random number between $(0, 1)$ and $|Z_\alpha|$ is the absolute value of Z-score (standard score) of α.

Before discussing the results, it is worth noting that some preliminary simulations were performed to see whether the mean order size, μ, has any effect on the solution quality of the order picking decisions and if so, to incorporate that as another factor in item classifications (along with order's CV and frequency). So, two levels of μ, namely *small* and *large* order size, were considered as another factor in classifying items. The results of simulations showed that mu does not have a significant effect on item classification and thus, the quality of rules. In other words, rules do not need to be customized to reflect the effect of average order size.

This result may seem counter-intuitive as one thinks, for example, it is better to pick the higher volume orders (count-wise and not weight-wise) from the closer locations to reduce the number of trips and thus, the distance objective value. This is true when either the whole order picking cannot be completed in one trip or when multiple orders are picked simultaneously. However, as explained in Section II.A, one important assumption of this research is considering *single order picking* operation in the warehouse. So, regardless of the order volume and its location, only one order is

picked in each trip and it can be picked fully.

Each of the randomly generated instances of the multi-objective order picking problem, are first solved with PAES algorithm. This provides the approximate nondominated front of each item. Then depending on the item class, different rules are applied. Having both PAES and the rules' results, the solution quality and the average gap for each individual item in each class can be compared. Next section discusses these results.

2. Results

To obtain the results of multi-objective optimization for each item class, 30 independent runs of PAES algorithm, each with 10000 iterations with the archive size of 100 was performed (The size of PAES archive was set to 100 which means up to 100 nondominated points are stored). Then, one customized rule per item class (based on Tables 11–14) was applied. Table 15 shows the hypervolumes of PAES and the rules and the average percentage gap between the two. Note that column "Average Gap" in Table 15 is the best average percentage gap of the 30 independent runs of PAES from the rules. Columns "Min Gap" and "Max Gap" are the minimum and maximum gap, respectively. A negative value indicates that either the rule's point is another point in the approximate nondominated front or it is in a higher (better) front and thus, it increases the value of hypervolume and makes the gap negative. Recall that the result of multi-objective optimization was an approximation.

The results in Table 15 shows that the customized rules, on average, can provide good quality points with an acceptable gap from the PAES approximate nondominated front. Figure. 11 depicts these results in a boxplot. To determine the effects of variability and frequency levels on the gap, a two-way analysis of variance (ANOVA) was conducted. As seen in Table 16, variability is the only significant factor in $\alpha = 0.05$ level (p-value < 0.05). The ANOVA was redone with the variability as

TABLE 15
PERCENTAGE GAP BETWEEN HYPERVOLUMES OF PAES AND THE RULES

Item Class (Variability, Frequency)	**Average Gap (%)**	**Min Gap (%)**	**Max Gap (%)**
Low, Low	7.24	-10.89	80.23
Low, High	14.27	-4.90	71.61
High, Low	-0.42	-8.74	6.58
High, High	-0.001	-8.30	4.13

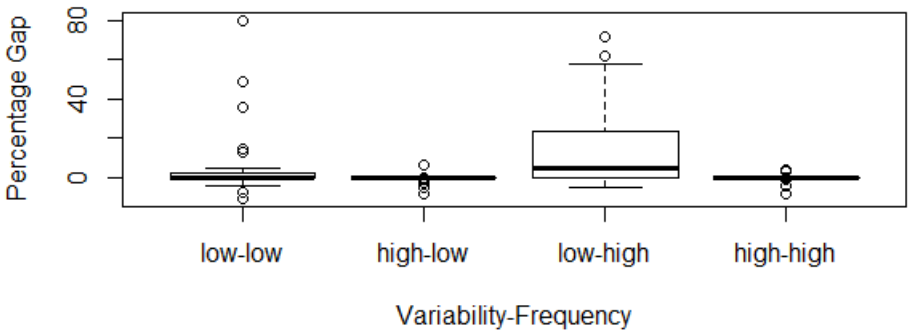


FIGURE 11 – Boxplot of variability and frequency effects on percentage gap

the only factor. The results are shown in Table 17.

Although the interaction of variability and frequency was insignificant, the interaction plot shown in Figure. 12 illustrates the effects of their level changes on the response (percentage gap). It is seen from Figure. 12 and the ANOVA results that in the *items with higher order variability*, using rules can create competitive solution(s) with smaller gaps, thus closer, to the approximate nondominated frontier.

It is worth noting that seeing bigger gaps for items with lower order variability relates to the inventory amount in a location. Due to lower variability, it may take longer for these items to transfer from a *low* inventory level to an *average* or *exact* level. This can cause picks from batches with later expiration dates and farther distances while the location is used for a longer period, thus higher space cost. However, this

TABLE 16
TWO-WAY ANOVA FOR THE EFFECTS OF VARIABILITY AND FREQUENCY ON GAP

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
variability	1	0.29673	0.29673	14.9117	0.0001915
frequency	1	0.04455	0.044553	2.2389	0.1374623
variability:frequency	1	0.03835	0.03835	1.9272	0.1678943
Residuals	109	2.169	0.019899		

TABLE 17
ANOVA FOR THE EFFECTS OF VARIABILITY ON GAP

	Df	**Sum Sq**	**Mean Sq**	**F value**	**Pr(>F)**
variability	1	0.29673	0.29673	14.626	0.0002169
Residuals	111	2.25191	0.020287		

is the case when replenishment is not instantaneous nor a new SKU is replaced the current one. Otherwise, picking from a location with *low* inventory level is equally appealing.

B. Real-World Data

To test and validate the performance of the rules, they were applied on a set of real-world data, supplied by one of UPS-Supply Chain Solutions (SCS) health-care compliant facilities, located in Louisville, KY. UPS-SCS, headquartered in Alpharetta, GA, offers transportation and freight services, logistics and distribution, consulting, and customs brokerage services and industry solutions (*UPS Supply Chain Solutions*).

The data set is from a healthcare warehouse. In this dataset, the levels of the storage locations (*close, middle range, far*) are defined not only based on the physical locations, but also the difficulty in accessibility. So, a location on a higher shelf which requires additional tools for order picking, is harder to access and thus, considered *far* (out of reach) in distance objective function. For confidentiality purposes, all the names are replaced with arbitrary letters.

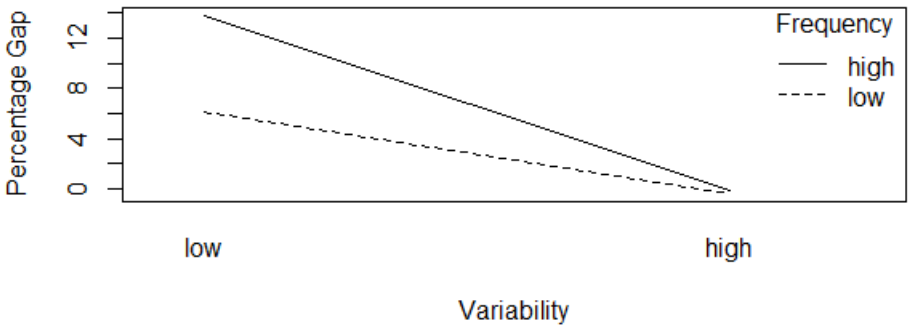


FIGURE 12 – Interaction plot between variability and frequency

1. Healthcare Warehouse

Two files were provided for this case study; "shipping velocity" and "inventory balance". The order quantity and picking location for each date was extracted from the shipping velocity file and all the other information regarding the location, batch, expiration date, and inventory were derived from the "inventory balance" file. The distance of each location from the I/O was calculated based on the floor layout.

The dataset was consisted of two weeks of data for 94 SKUs where 67 of them are high order frequency SKUs (more than one order per day) which are the focus of the evaluation. 60 of these SKUs are stored in at least two areas: One in fast-picking area and one in reserved area. All the SKUs are from one zone, i.e. they belong to the same client. Orders are first picked from fast-picking area and as soon as the location becomes empty, replenishment is triggered. Because FEFO policy is practiced, expiration date objective has a high priority. If the order is for full pallet, it is picked from the reserved area. This can be considered as the *exact* inventory level in the rules.

The distribution of the order size is required for the analysis. For most of the SKUs the order size was very skewed (high variability) and transformation of the data

was necessary. Assuming x_i is a random variable representing the order size of SKU i, $y_i = \log_{10}(x_i)$ followed a normal distribution.

Practicing the FEFO policy and immediate replenishment of the closer area is equivalent to keeping the batch with *early* expiration date *close* to the I/O, namely (*early, close, high*) state. Also, picking full pallets from reserved area is the same as choosing a location with *exact* inventory level regardless of its distance or expiration date. In a nutshell, expiration date has the highest priority and then comes distance. Inventory of a location only matters when it is *exact* and can completely fulfill the order. The rules in Table 18 capture these preferences.

One can see from Table 18 that only states with *exact* inventory level are preferred to the (*early, close, high*) state (*average* and *low* inventory levels happen by picking from a *high* inventory location repeatedly). Because the orders have high variability and frequency, comparing the sequence of rules in Table 18 with those of Table 14 can highlight the situations that could be improved. Keeping the priority order (expiration date, distance, inventory), one can conclude that the only states which changing their sequence in Table 18 could potentially enhance the order picking performance, are those with *low* inventory level.

So, as long as an empty location in closer areas is not replenished with the exact same batch from the reserved area (i.e. the same SKU with the same expiration date), the order picking performance (i.e. objective function values) could be improved by **not** picking the batch with *low* inventory level. Because the variability is high, these locations would eventually transform to an *exact* level. Table 19 show the sequence of rules in such scenarios.

The rules from both Tables 18 and 19 were applied on 60 SKUs of the health-care warehouse dataset. In all but one SKU, both tables generated the same results. The exception happened for an SKU with similar situation as mentioned above (not replenishing the closer location with the exact same batch). In that instance, rules

TABLE 18
SCS RULES

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Early	Middle Range	Exact	0.944	2
Early	Far	Exact	0.89	3
Middle	Close	Exact	0.944	4
Middle	Middle Range	Exact	0.838	5
Middle	Far	Exact	0.738	6
Late	Close	Exact	0.89	7
Late	Middle Range	Exact	0.738	8
Late	Far	Exact	0.6	9
Early	Close	Low	0.838	10
Early	Close	Average	0.944	11
Early	Close	High	0.89	12
Early	Middle Range	Low	0.644	13
Early	Middle Range	Average	0.838	14
Early	Middle Range	High	0.738	15
Early	Far	Low	0.476	16
Early	Far	Average	0.738	17
Early	Far	High	0.6	18
Middle	Close	Low	0.644	19
Middle	Close	Average	0.838	20
Middle	Close	High	0.738	21
Middle	Middle Range	Low	0.333	22
Middle	Middle Range	Average	0.644	23
Middle	Middle Range	High	0.476	24
Middle	Far	Low	0.124	25
Middle	Far	Average	0.476	26
Middle	Far	High	0.271	27
Late	Close	Low	0.476	28
Late	Close	Average	0.738	29
Late	Close	High	0.6	30
Late	Middle Range	Low	0.124	31
Late	Middle Range	Average	0.476	32
Late	Middle Range	High	0.271	33
Late	Far	Low	0	34
Late	Far	Average	0.271	35
Late	Far	High	0.071	36

TABLE 19
IMPROVED SCS RULES

Expiration Date	**Distance**	**Inventory**	**Confidence**	**Order**
Early	Close	Exact	1	1
Early	Middle Range	Exact	0.944	2
Early	Far	Exact	0.89	3
Middle	Close	Exact	0.944	4
Middle	Middle Range	Exact	0.838	5
Middle	Far	Exact	0.738	6
Late	Close	Exact	0.89	7
Late	Middle Range	Exact	0.738	8
Late	Far	Exact	0.6	9
Early	Close	Average	0.944	10
Early	Close	High	0.89	11
Early	Middle Range	Average	0.838	12
Early	Middle Range	High	0.738	13
Early	Far	Average	0.738	13
Early	Far	High	0.6	15
Middle	Close	Average	0.838	16
Middle	Close	High	0.738	17
Middle	Middle Range	Average	0.644	18
Middle	Middle Range	High	0.476	19
Middle	Far	Average	0.476	19
Middle	Far	High	0.271	21
Late	Close	Average	0.738	22
Late	Close	High	0.6	23
Late	Middle Range	Average	0.476	24
Late	Middle Range	High	0.271	25
Late	Far	Average	0.271	25
Late	Far	High	0.071	27
Early	Close	Low	0.838	28
Early	Middle Range	Low	0.644	29
Early	Far	Low	0.476	30
Middle	Close	Low	0.644	31
Middle	Middle Range	Low	0.333	32
Middle	Far	Low	0.124	33
Late	Close	Low	0.476	34
Late	Middle Range	Low	0.124	35
Late	Far	Low	0	36

of Table 18 emptied the location one day faster than those of Table 19 but its total traveled distance was 8.4% higher (2435.4 feet compared to 2230.2 feet). The reason for this was different orders of (*early, close, low*) and (*early, far, high*) states. In the rules shown in Table 18, the orders are 10 and 18, respectively and in Table 19 they are 15 and 28, respectively.

Although just 8.4% improvement in total traveled distance of only one SKU out of 60, is negligible, but the potential effects of implementing a new rule cannot be denied. Note that, the data was for two weeks, one product class (high variability and high order frequency) and from one zone. It is speculated that more comprehensive analysis on much bigger datasets can reveal more potential benefits of applying these simple rules.

C. Conclusion

Much research has been done in the area of warehouse order picking. Some of the well-studied problems in this area are (De Koster, Le-Duc, and Roodbergen 2007):

Storage location assignment problem and effects of implementing different storage policies (such as random, dedicated, class-based) on order picking;

Forward-reserve allocation problem to find out which SKUs and in what quantity should be stored in the fast-picking areas for more efficient order picking; and

Routing problem to find the best sequence of picking items on a pick list or routing the order pickers in a warehouse.

Jointly considering multiple warehousing functions and studying their interactions and effects on order picking operations has also received attention in the last decade but not many studies have been done regarding multiple criteria decision making for order picking operations in healthcare warehouses. This research aimed to fill this gap by finding the best picking location to minimize travel distance, spoilage,

space usage, and maximize future order's fulfillment chance.

It was discussed that solving the multi-objective problem and presenting one single solution to the decision maker does not provide the full picture. Also, providing all the solutions is not always achievable as it is computationally expensive and most of the times, a set of rules and regulations drive decision makers toward the chosen solution. Therefore, a novel approach based on association rule mining was presented to categorize the solutions obtained from the multi-objective optimization, recognize a common pattern from them, and generate order picking strategies based on the patterns and the preferences of the decision maker.

Although initial storage assignment were not of concern in this research, it is clear that slotting decisions in the warehouse has a direct effect on the order picking rules. What will happen to the empty locations can influence from which location the order should be picked. Is the empty location replenished with the same SKU or another SKU will occupy that location? If dynamic slotting is of interest to the managers and a location should be empty for a new SKU, then depleting a location faster and making it available is preferred. In this situation a state such as (*far, average*) can be preferred over (*close, high*). Otherwise and in case of replenishing the SKU, picking from the closer and more efficient state is preferred.

Finally, for inexpensive items it might be acceptable to move the insufficient remainder to another location to empty the space for more valuable items or even move them to some shared space with all the other inexpensive remainders (assuming the location does not require to have special conditions), especially, if the item's order frequency is low. But if such moves are not possible or costly, the importance of location's inventory becomes more noticeable.

The results obtained from all the rules (Tables 11–14 and 18–19), boxplot in Figure. 11, and the ANOVA table are summarized below:

- The rules evolved from classification of SKUs and the demand profile can create

good quality solution on average.

- The best quality results are obtained when the orders have *high variability*.
- When the order variability is low, the solutions are more spread out (more low quality solutions).
- Stick to the FEFO rule and pick from the locations based on their batches' expiration dates. Unless the inventory is either in the *exact* or *low* level.
- Regardless of the expiration date and the distance, it is recommended to pick from a location with *exact* inventory level and empty the space.
- If replenishment is not instantaneous, it is recommended to **not** pick from a location with *low* inventory level, regardless of its expiration date and the distance. It is better to wait until the inventory in that location transfers to an *average* or *exact* level.
- If fast replenishment is obtainable, prioritize distance over inventory objective.

The points generated by these rules are either in the approximate nondominated frontier or, on average, within an acceptable gap (for high variability orders, this gap is less than $\pm 8.8\%$). It is worth noting that in a healthcare warehouse with well-organized areas, the batches with earlier expiration dates are in the closer areas which means two of the objectives, namely, close distance and early expiration date, are already satisfied and the decisions narrow down to the location's inventory and space usage.

D. Future Work

Two main paths can be continued from this research: 1) Work on improvement of multi-objective optimization algorithms and also generating better rules; 2)

Work on combining other warehouse operations with order picking and applying this research on other types of warehouses.

1. Improvement of MOO Algorithms and The Rules

The neighborhood solution generator of PAES can be improved. Other heuristics such as variable neighborhood search can be introduced to have a hybrid method. Also, rules can be used interactively with any MOO algorithm. So, in each iterations and for finding a candidate solution from the current solution, the rules can be used. In other words, instead of randomly choosing a new batch-location for the candidate solution in each iteration, it is possible to generate the solution based on the rules (for example state of (*early, middle range, high*).

In this research, (1+1)-PAES was used. Other forms of PAES such as $(1+\lambda)$-PAES can be implemented. By doing that, other types of neighborhood solution generators (such as crossover of two solutions with repair procedure) can be explored. Also, the quality of other MOO algorithms such as multi-objective particle swarm optimization, on this type of problem structure can be investigated.

Regarding the rules, three levels of expiration date and distance, and four levels of inventory were used in classifying the SKUs. Other levels can be introduced here to see their effects on solution quality. For example, instead of dividing expiration date into three categories, a threshold can be introduced to serve as a lowerbound. Then items are categorized based on the closeness of their expiration date to this bound. A new level can also be defined as "alarming" and if a batch gets close or reaches that level, it has a priority over all the other criteria and the pick must be done from that batch.

2. Other Warehouse Operations

Single order picking was addressed in this research. An interesting and practical next step would be to investigate the *batch picking* where multiple orders are picked simultaneously. Here more data training for the rules are necessary as the relations between different item classes and their storage locations become significantly important. The power of association rule mining is revealed better in these cases.

It is very clear that the initial storage assignment of items in the warehouse matter and they affect the efficiency of order picking. The location assignment decisions can be combined with the order picking strategies. For example, if an SKU has a high order frequency, then it should be stored in closer areas to increase order picking efficiency. Also in a single order picking warehouse, if a batch occupies more than one location, it is better that the locations are close together. So, if the order needs to be partially picked from one location, the other one is close enough. However this might not be the case in a batch picking warehouse. Because depending on the relation of different SKUs together, it might be preferred not to centralized batches of an SKU.

To deal with such situations, a new criteria called "*extra trip*" can be introduced. So for example, if instead of picking an order from a location in (*early, close, high*) state, the pick happens from (*early, middle range, high*) state, the *extra trip* value can increase to trigger the inefficiency in storage assignment. This can also occur if the order is partially picked from multiple locations.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases". In: *Acm sigmod record*. Vol. 22. 2. ACM. 1993, pp. 207–216.

[2] Rakesh Agrawal and Ramakrishnan Srikant. "Fast algorithms for mining association rules". In: *Proc. 20th int. conf. very large data bases, VLDB*. Vol. 1215. 1994, pp. 487–499.

[3] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb. "A simulated annealing-based multiobjective optimization algorithm: AMOSA". In: *IEEE transactions on evolutionary computation* 12.3 (2008), pp. 269–283.

[4] John J Bartholdi III and Steven T Hackman. "Warehouse & distribution science: release 0.96". In: (2014).

[5] Jeroen P Van den Berg, Gunter P Sharp, AJRM Noud Gademann, and Yves Pochet. "Forward-reserve allocation in a warehouse with unit-load replenishments". In: *European Journal of Operational Research* 111.1 (1998), pp. 98–113.

[6] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. "A criterion space search algorithm for biobjective mixed integer programming: The triangle splitting method". In: *INFORMS Journal on Computing* 27.4 (2015), pp. 597–618.

[7] Natashia Boland, Hadi Charkhgard, and Martin Savelsbergh. "A new method for optimizing a linear function over the efficient set of a multiobjective integer program". In: *European Journal of Operational Research* (2016).

[8] Lucas Bradstreet. *The hypervolume indicator for multi-objective optimisation: calculation and use*. University of Western Australia, 2011.

[9] Karl Bringmann and Tobias Friedrich. "Approximating the volume of unions and intersections of high-dimensional geometric objects". In: *International Symposium on Algorithms and Computation*. Springer. 2008, pp. 436–447.

[10] Felix TS Chan and Hing Kai Chan. "Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage". In: *Expert Systems with Applications* 38.3 (2011), pp. 2686–2700.

[11] Hadi Charkhgard. "Theory and algorithms for multi-objective integer programming| NOVA. The University of Newcastle's Digital Repository". In: (2016).

[12] Carlos A Coello Coello, Gregorio Toscano Pulido, and M Salazar Lechuga. "Handling multiple objectives with particle swarm optimization". In: *IEEE Transactions on evolutionary computation* 8.3 (2004), pp. 256–279.

[13] Carlos Coello Coello, Gary B Lamont, and David A Van Veldhuizen. *Evolutionary algorithms for solving multi-objective problems*. Springer Science & Business Media, 2007.

[14] R De Koster. "How to assess a warehouse operation in a single tour". In: *Technology Report. Erasmus University, Netherlands* (2004).

[15] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. "Design and control of warehouse order picking: A literature review". In: *European Journal of Operational Research* 182.2 (2007), pp. 481–501.

[16] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Vol. 16. John Wiley & Sons, 2001.

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[18] J. Paul Dittmann. *Best practices for managing cost in the healthcare supply chain*. UPS. URL: `https://www.ups.com/media/en/best-practices-managing-cost-part1.pdf` (visited on 10/30/2016).

[19] Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

[20] *Eighth UPS Pain in the Chain Survey*. UPS, 2015. URL: `https://www.ups.com/media/en/UPS-PITC-Executive-Summary-North-America.pdf` (visited on 10/30/2016).

[21] Mark Fleischer. "The measure of Pareto optima applications to multi-objective metaheuristics". In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer. 2003, pp. 519–533.

[22] Edward H Frazelle, Steven T Hackman, U Passy, and Loren Kerry Platzman. *The forward-reserve problem*. John Wiley & Sons, Inc., 1994.

[23] Xavier Gandibleux. *Multiple criteria optimization: state of the art annotated bibliographic surveys*. Vol. 52. Springer Science & Business Media, 2006.

[24] Jinxiang Gu, Marc Goetschalckx, and Leon F McGinnis. "Research on warehouse design and performance evaluation: A comprehensive review". In: *European Journal of Operational Research* 203.3 (2010), pp. 539–549.

[25] Jinxiang Gu, Marc Goetschalckx, and Leon F McGinnis. "Research on warehouse operation: A comprehensive review". In: *European journal of operational research* 177.1 (2007), pp. 1–21.

[26] Steven T Hackman, Meir J Rosenblatt, and John M Olin. "Allocating items to an automated storage and retrieval system". In: *IIE transactions* 22.1 (1990), pp. 7–14.

[27] Michael Hahsler, Bettina Gruen, and Kurt Hornik. "arules - A Computational Environment for Mining Association Rules and Frequent Item Sets". In: *Journal of Statistical Software* 14.15 (2005), pp. 1–25. ISSN: 1548-7660. URL: `http://dx.doi.org/10.18637/jss.v014.i15`.

[28] Pierre Hansen and Nenad Mladenović. "Variable neighborhood search: Principles and applications". In: *European journal of operational research* 130.3 (2001), pp. 449–467.

[29] C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*. Vol. 164. Springer Science & Business Media, 2012.

[30] Elizabeth Jewkes, Chulung Lee, and Ray Vickson. "Product location, allocation and server home base location for an order picking line with multiple servers". In: *Computers & Operations Research* 31.4 (2004), pp. 623–636.

[31] Dylan F Jones, S Keyvan Mirrazavi, and Mehrdad Tamiz. "Multi-objective meta-heuristics: An overview of the current state-of-the-art". In: *European journal of operational research* 137.1 (2002), pp. 1–9.

[32] João Pedro Jorge, Zafeiris Kokkinogenis, Rosaldo JF Rossetti, and Manuel AP Marques. *Simulation of an order picking system in a pharmaceutical warehouse*. 2012.

[33] Boris Klots, William Henry Waddington, Patricia C Grewell, Peter Ham, Susan L Griese, and Gerry Perham. *Order allocation to select from inventory locations stocking few units of inventory*. US Patent 6,622,127. 2003.

[34] Joshua D Knowles and David W Corne. "Approximating the nondominated front using the Pareto archived evolution strategy". In: *Evolutionary computation* 8.2 (2000), pp. 149–172.

[35] Marco Laumanns, Günter Rudolph, and Hans-Paul Schwefel. *Approximating the pareto set: Concepts, diversity issues, and performance assessment*. Secretary of the SFB 531, 1999.

[36] R Timothy Marler and Jasbir S Arora. "Survey of multi-objective optimization methods for engineering". In: *Structural and multidisciplinary optimization* 26.6 (2004), pp. 369–395.

[37] Nenad Mladenović and Pierre Hansen. "Variable neighborhood search". In: *Computers & Operations Research* 24.11 (1997), pp. 1097–1100.

[38] BALÁZS Molnar and GYÖRGY Lipovszki. "Multi-objective routing and scheduling of order pickers in a warehouse". In: *International Journal of Simulation* 6.5 (2002), pp. 22–32.

[39] Jason Chao-Hsien Pan and Po-Hsun Shih. "Evaluation of the throughput of a multiple-picker order picking system with congestion consideration". In: *Computers & Industrial Engineering* 55.2 (2008), pp. 379–389.

[40] Charles G Petersen and Gerald Aase. "A comparison of picking, storage, and routing policies in manual order picking". In: *International Journal of Production Economics* 92.1 (2004), pp. 11–19.

[41] PN Poulos, GG Rigatos, SG Tzafestas, and AK Koukos. "A Pareto-optimal genetic algorithm for warehouse multi-objective optimization". In: *Engineering Applications of Artificial Intelligence* 14.6 (2001), pp. 737–749.

[42] Suk-Chul Rim and In-Sun Park. "Order picking plan to maximize the order fill rate". In: *Computers & Industrial Engineering* 55.3 (2008), pp. 557–566.

[43] Claudio Fabiano Motta Toledo, Paulo Morelato França, Reinaldo Morabito, and Alf Kimms. "Multi-population genetic algorithm to solve the synchronized and integrated two-level lot sizing and scheduling problem". In: *International Journal of Production Research* 47.11 (2009), pp. 3097–3119.

[44] James A Tompkins, John A White, Yavuz A Bozer, and Jose Mario Azaña Tanchoco. *Facilities planning*. John Wiley & Sons, 2010.

[45] *UPS Supply Chain Solutions*. URL: `https://www.ups-scs.com/about/` (visited on 10/10/2016).

[46] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. "A faster algorithm for calculating hypervolume". In: *IEEE transactions on evolutionary computation* 10.1 (2006), pp. 29–38.

[47] Qingfu Zhang and Hui Li. "MOEA/D: A multiobjective evolutionary algorithm based on decomposition". In: *IEEE Transactions on evolutionary computation* 11.6 (2007), pp. 712–731.

[48] Aimin Zhou, Bo-Yang Qu, Hui Li, Shi-Zheng Zhao, Ponnuthurai Nagaratnam Suganthan, and Qingfu Zhang. "Multiobjective evolutionary algorithms: A survey of the state of the art". In: *Swarm and Evolutionary Computation* 1.1 (2011), pp. 32–49.

[49] Eckart Zitzler. "Evolutionary algorithms for multiobjective optimization: Methods and applications". In: (1999).

[50] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. "SPEA2: Improving the strength Pareto evolutionary algorithm". In: *Eurogen*. Vol. 3242. 103. 2001, pp. 95–100.

[51] Eckart Zitzler and Lothar Thiele. "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach". In: *IEEE transactions on Evolutionary Computation* 3.4 (1999), pp. 257–271.

[52] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. "Performance assessment of multiobjective optimizers: an analysis and review". In: *IEEE transactions on evolutionary computation* 7.2 (2003), pp. 117–132.

CURRICULUM VITAE

Ehsan Khodabandeh email: e.khodabandeh@gmail.com

Education:

PhD, Industrial Engineering, University of Louisville, 2016

MS, Industrial Engineering, Sharif University of Technology, Tehran, Iran, 2011

BS, Industrial Engineering, Iran University of Science and Technology, Tehran, Iran, 2007

Work Experience:

Operations Research Scientist, Opex Analytics, 2016

Operations Research Intern, Norfolk Southern Corporation, 2015

Lecturer, IE 380 Work Design, Department of Industrial Engineering, University of Louisville, 2013–2014

Operations Research Analyst, GE Appliances & Lighting, 2012–2013

Computer Skills:

Programming Languages: Visual C#, Python, MATLAB

Technical Software: CPLEX, Pulp, Minitab, R, ARENA, AnyLogic, PostgreSQL, Tableau

Activities and Awards:

Alpha Pi Mu, Industrial Engineering Honorary Society, University of Louisville, Member

Recipient of Material Handling Education Foundation Inc. Honor Scholarship, 2016

Outstanding Industrial Engineering Graduate Student Award, University of Louisville, 2014

Graduate Student Council (GSC), University of Louisville, Industrial Engineering Representative, 2013–2015

International OASIS, a multicultural student organization, University of Louisville, President,

2013–2016

Publication:

Khodabandeh, E., Bai, L., Heragu, S. S., Evans, G. W., Elrod, T., and Shirkness, M. Modelling and Solution of a Large-Scale Vehicle Routing Problem at GE Appliances & Lighting. International Journal of Production Research, doi=http://dx.doi.org/10.1080/00207543.2016.1220685

Haji, R., Khodabandeh, E., and Haji, A. (2011). Rework Center Attached to a Queueing-Inventory System with Budgetary Constraint. In Mechanical, Industrial, and Manufacturing Engineering Proceedings of 2011 International Conference on Mechanical, Industrial, and Manufacturing Engineering (MIME 2011).

Haji, R., Haji, A., and Khodabandeh, E. (2011). Queueing-Inventory Model with Rework, Backorders, and Poisson Demand. In Production and Operations Management Society (POMS) 22nd Annual Conference, Reno, Nevada, USA, April (Vol. 29).