A DATA-INTENSIVE ANALYSIS AUGMENTED SIMULATION MODEL OF AN ORDER PICKING OPERATION

by

Yue Li, B.S.

A thesis submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Master of Science
with a Major in Engineering
August 2017

Committee Members:

Jesus A. Jimenez, Chair

Eduardo Perez, Co-Chair

Francis A. Méndez Mediavilla

DEDICATION

I would like to say thank you to Linda Anderson, who is my good friend and ESL teacher in Columbus, Ohio for her insightful suggestions and corrections on this thesis. As a teacher, she passes her knowledge, experience as well as her positive life attitude to the young people like me. She is always willing to give me a hand and trying to encourage me whenever I’m up the creek. I think I am the luckiest one to have such a friend and teacher like her. Thank you, Linda.

ACKNOWLEDGEMENTS

During my graduate studies in Texas State University, several faculties and staffs collaborated with my research directly or indirectly. It would be impossible for me to fulfill my graduate study without their support and assistant.

Foremost, I offer my sincerest gratitude to my advisor, Dr. Jesus A. Jimenez, who has supported and advised me with his knowledge and patience throughout my thesis. This thesis would not have been completed without his time and effort.

Besides my advisor, I would like to thank the rest of my thesis committee members: Dr. Eduardo Perez, Dr. Francis A. Méndez Mediavilla for their insightful guidance, suggestion and comments.

Also, I wish to thank all the IE staff in Ingram School of Engineering for their kindness and patience throughout my graduate.

Finally, I would like to thank my family – my wife Nan Cheng, my son Anthony C Li and my daughter Sophia C Li, for their unconditional support, love, and inspiration.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	**Description**
ARM	Association Rules Mining
DC	Distribution Center
CAPM	Clustering-Assignment Problem Model
OOS	Order Oriented Slotting
I/O	Input/output
ILP	Integer Linear Programming
COI	Cube-per-Order Index
SKU	Stock Keeping Unit
MLI	Maximum Loop Insertion
MTLI	Minimum Travelling Loop Insertion
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem
NP-hard	Non-deterministic Polynomial-time hard

ABSTRACT

Order picking is the most labor-intensive function of distribution centers (DC) in the food and beverage store industry. An efficient order picking process supports this industry’s supply chain to move high volumes of products between the DC and the retail stores. This thesis focuses on the storage location assignment problem to deciding via an algorithm based on Association Rules Mining (ARM) the most adequate location of incoming products. The algorithm analyzes hundreds of orders received by the DC to find correlated products that are ordered frequently together by retail stores. The algorithm then assigns correlated products to storage locations that are close to each other in order to minimize order picking times. The results of computer simulation experiments using data from a real distribution center will be presented to evaluate the performance of the DC layout resulting from ARM.

Keywords: Simulation, Data-intensive Analysis, Distribution Center, Facilities Layout, Order Picking, Association Rules.

I. INTRODUCTION

1.1 Problem Description

In order to meet the requirements of demand-driven markets and maintain the satisfaction level of customers, both managers and researchers seek to save the cost and time in the supply chain. In supply chain management, the distribution center (DC) plays a significant intermediate role between suppliers and customers. Receiving, storing, order picking, sorting and distributing are the basic activities of a DC. Order picking is the most time-consuming and labor-intensive process [1], and it approximately accounts for 55% of warehouse operating expenses [2]. By optimizing the order picking process, time and cost can be reduced, thus, increases the DC's efficiency.

The efficiency of a DC determines the performance of the whole supply chain and the competitiveness of the firm. The challenges regarding the distribution center are not only managing massive amounts of goods and items, but also delivering thousands of daily orders in a timely manner. The storage location assignment method targets the efficient and systematic allocation of Stock Keeping Units (SKUs) to warehouse slots in order to minimize the order-picking time. [3] A typical requirement for storage location assignment solution is to ensure the SKUs are associated by specific metric, such as similarity, turnover ratio, product flow, and distance [1]. Due to the complexity of storage location assignment problem, a quick solution of storage location assignment problem and facility layout design in practical level becomes more important and valuable to warehousing operations in the supply chain. [3]

ARM algorithm has been widely used by the food and grocery industry [4][5] to create efficient layout designs of their retail stores. ARM is a data-mining technique which focuses on identifying the correlated products among different transactions. ARM algorithm shows the set of products that are frequently ordered together. These rules can be used for creating batches in layout design and products allocation in supermarkets, for example the well-known diaper-beer case [6], as well as pushing advertisements and issuing coupons to targeted customers to increase the company's sales. Positioning items in the distribution center locations is done heuristically by several companies. For example, they put items that are of similar weight near each other. In this research, the application of ARM is extended to the distribution center. The ARM algorithm is used to find the correlations between SKUs that are frequently ordered together among all the transactions from different retail stores. The correlated SKUs will be reassigned to locations closer to each other based on ARM algorithm as well as weight-classed policy. See Figure 1.

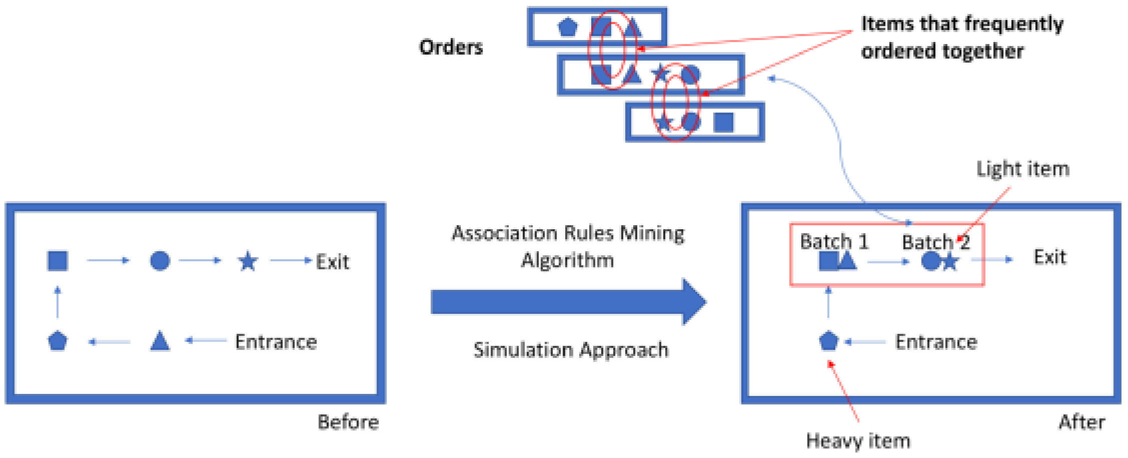


Figure 1 Illustration of using ARM algorithms to create batches

In order to improve the order picking efficiency in the DC, different types of technologies such as data mining, mathematical formulation, heuristic method as well as computer simulation, are implemented in this thesis to fill the gap from data collection to performance comparison. As is mentioned above, ARM algorithm identifies the correlated batches of products. However, such rules don't take locations into consideration. Therefore, the output of the ARM algorithm is used as input for exchange heuristic to solve location assignment problem to achieve the goal of assigning products into the best locations. The output of the exchange heuristic will be inputted into simulation model, which will provide the estimated travel distance, vehicle utilization, and order-picking time - the performance metrics used in the comparison of layouts.

1.2 Objectives

The objectives of this thesis are as follow:

a. Integrate ARM technique, mathematical programming and computer simulation to produce a solution for a Quadratic Assignment-based storage location problem.
b. Identify the best combination of support and confidence for the ARM algorithm in the context of the DC.
c. Design a heuristic to find solutions for the storage location problem in a practical level in term of computational time and difficulty of implementation.

1.3 Organization of Thesis

The organization of this thesis is as follows: Section 2, a review of the literature of warehousing and order picking activities. Section 3, the methodology from five aspects: weight-based class, ranking and sorting, ARM, exchange heuristic, and computer simulation. Section 4, the experiments and numerical results. Section 5, the conclusion and future work.

II. LITERATURE REVIEW

2.1 Warehousing

Storing or buffering products such as raw materials, goods-in-process, and finished products between suppliers and customers is referred to as "warehousing". Different terms are used to describe a warehouse when additional functions are added: "Distribution center" is commonly used for a warehouse with function of distribution; "transshipment", "cross-docking", and "platform" center refer to warehouses with functions of temporarily holding product and quick transfer [1]. Based on ELA/AT Kearney (2004) [7], the cost of warehousing involved in logistics process is about 20% of a company's total supply chain management cost. The workflow includes: receiving products from both suppliers and customers; reserving storage for products; picking, sorting and packing products according to orders received from retail stores; and distributing and shipping the orders to the customers. Figure 2 shows the warehousing functions and corresponding product flows.

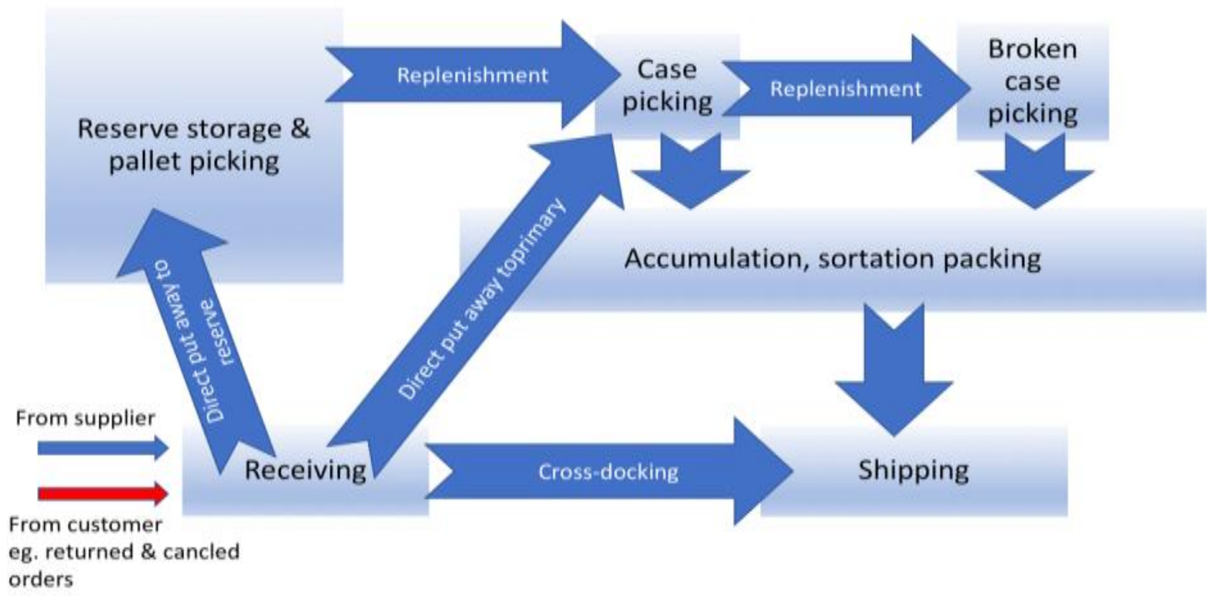

Reserve storage & pallet picking
Replenishment
Case picking
Replenishment
Broken case picking
Accumulation, sortation packing
Direct put away to reserve
Direct put away toprimary
From supplier
From customer eg. returned & cancled orders
Receiving
Cross-docking
Shipping

Figure 2 Typical warehouse functions and flows (based on Tompkins et al., 2003) [2]

De Koster *et al.* [1] specified that the factors that need to be considered to enhance warehousing performance are the number of storage zones and main aisles as well as aisle dimensions.

2.2 Order Picking

In practice, multiple order-picking methods may be applied to a warehouse at the same time. These methods include picker-to-parts, parts-to-picker, sorting system, and pick-to-box [8]. On average, order picking accounts for 55% of warehouse operating expenses. Among all the activities in order picking, travel consumes 50% of order picking time [2]. See Figure 3.

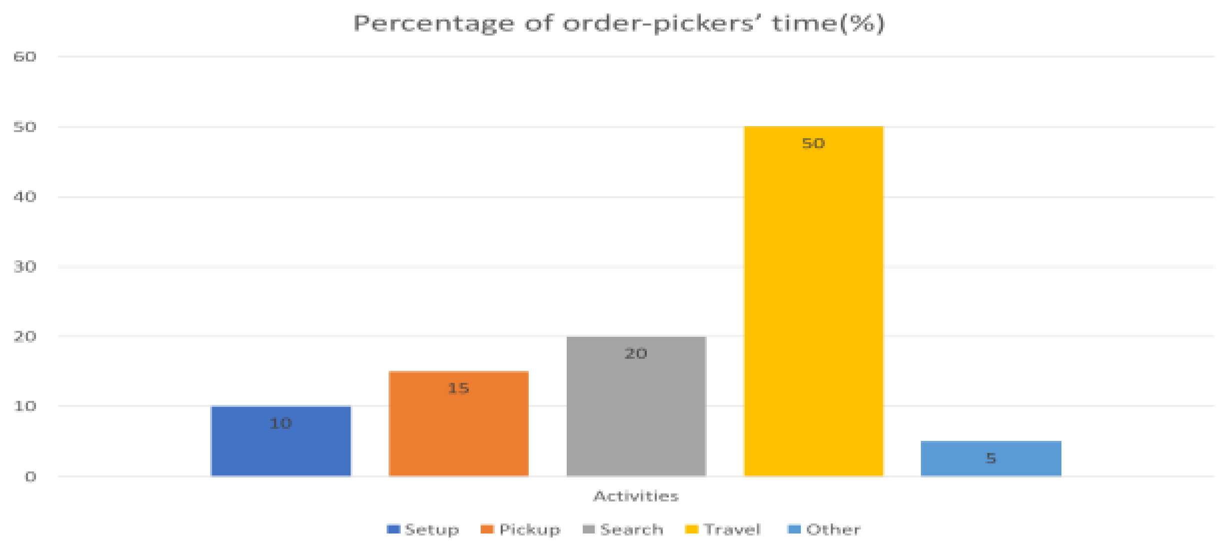


Figure 3 Percentage of order-picker's time (based on Tompkins et al., 1996)

De Koster *et al.* [1] introduced two branches of order picking methods –manual and automated picking– which are determined by the involvement of automated machines in picking process. See Figure 4 for details.

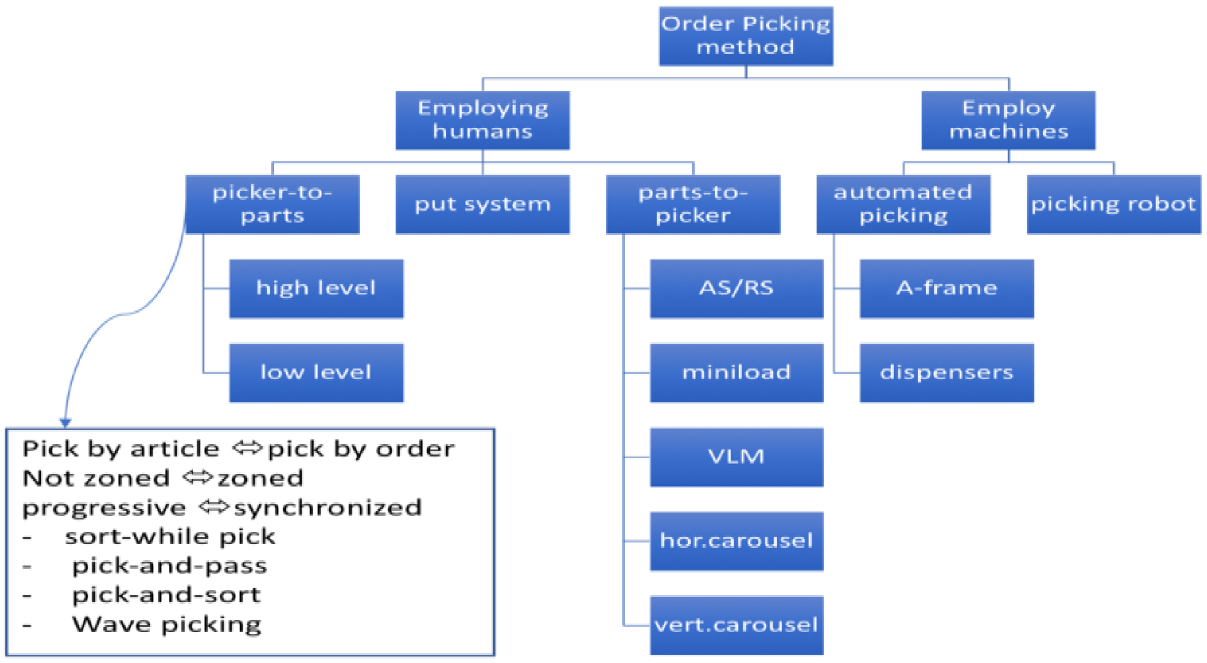


Figure 4 Order picking methods based on [1]

The picker-to-parts method is the most commonly implemented in real applications. In this method, pickers travel between the aisles to pick up the SKUs. The picker-to-parts method is classified into low level and high level order-picking system. In the low-level order picking system, the picker picks up the SKUs located on ground level of the bins. In the high-level order picking system, also known as man-aboard order picking system, the picker needs to stop at the picking location and lift the picking truck or crane to get the SKUs located at higher position. More time is required since the picker needs to operate or wait for the equipment to finish the picking process.

Some other classifications include batch picking. In batch picking, multiple orders are picked continuously. Wave picking is a term used if all the orders are released at the same time for picking in different warehouse areas, but they have a common destination after the pick up. According to Petersen (2000) [9], the required time to complete picking the whole batch is often between 30 minutes to 2 hours.

According to De Koster *et al*. [1], most academic research focuses on high-level and AS/RS order picking systems. This thesis concentrates in low-level order picking operations such as the picker-to-parts method.

2.3 Routing Heuristics

Routing is another important way to shorten the travel distance and order picking time. Ratliff and Rosenthal [10] proposed Optima, which is a routing strategy using dynamic programming. Optima repeats the procedures of determining the optimal order picking priority within each aisle and concerning the next aisle to find a shortest picking path. Hall [11], Petersen [12] and Roodbergen [13] proposed a few heuristic methods for routing selection in single-block warehouses such as the S-shape, return, mid-point, largest gap, combined and optimal routing heuristics. See details in Figure 5.

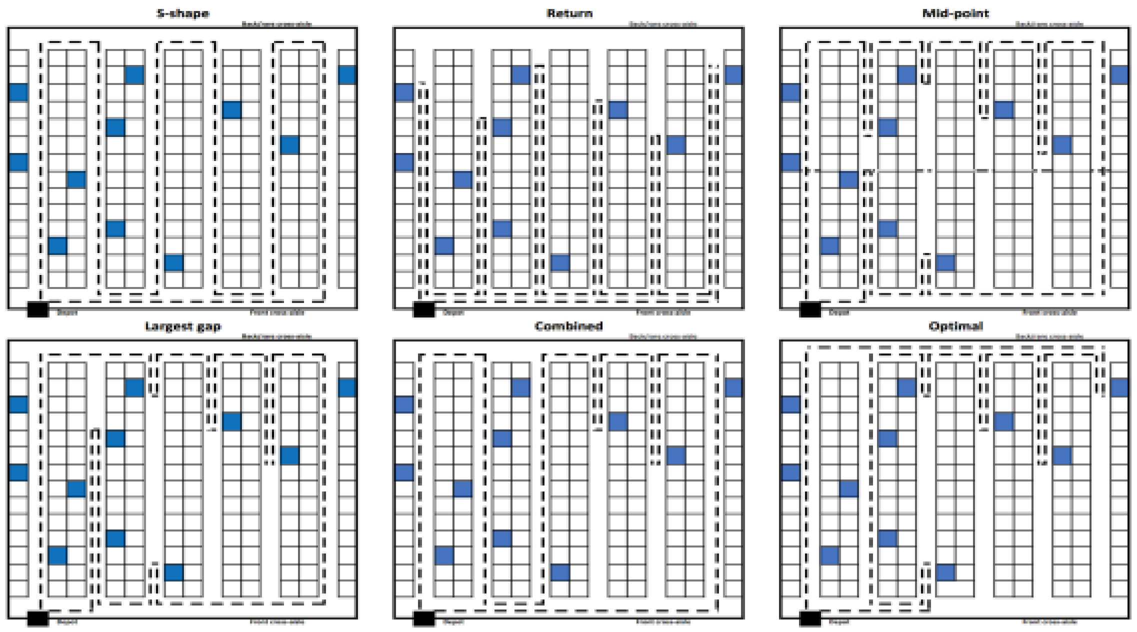


Figure 5 Example of routing methods for a single-block warehouse (based on [8] [11] [13]).

The S-shape routing method is the simplest and most commonly used. The picker follows a S-shape route along the aisles to pick up items. The aisles without picking jobs are not visited. In the return routing method, the picker enters and exits each aisle at the same position. After picking up the item required, the picker exits the aisle at the enter point of by retracing the route entered. The midpoint routing method was developed after the return routing method. The picker follows the same rules as return routing method, but the picker only reaches a maximum distance equivalent to half length of the aisle and then returns to the entrance of the aisle. Hall's study shows that this method is better than S-shape when a small number of SKUs is required to be picked up in each aisle [14]. The largest gap routing method has a slight variation to the midpoint routing method since it reaches as far as the largest gap in an aisle. Usually, the largest gap routing method has better performance than the midpoint method, but it's more complicated to implement. In the combined routing method, the picker either exits the aisle at the enter point of this aisle by retracing the route entered or at the end of the aisle. Because of the complicated computation, dynamic programming may be necessary to determine the route [15].

When selecting a routing plan in practice, multiple factors need to be taken into consideration, for example, balancing the improvement and complexity of the designed system. Hsieh and Huang [16] introduced a routing heuristic, named Maximum Loop Insertion (MLI), which balances the routing selection complexity and the travel distance reduction. The MLI method provides shorter total travel distance than those obtained with the Minimum Travelling Loop Insertion (MTLI) and Particle Swarm Optimization (PSO). The travel distance of PSO algorithm decreases when MLI result is used as the initial solution.

In this thesis, S-shape routing method is selected because S-shape routing is static, which then helps to make comparisons of different layout configurations. Also, the S-shape routing reduces the complexity of the layout design and the storage location assignment problem.

2.4 Layout Design and Storage Assignment

2.4.1 Layout Design

Layout design has two sub-problems: the facility layout and the aisle configuration problems. The first problem focuses on locating different departments including receiving, picking, storage, sorting and shipping, etc. by considering the operational interactions between departments with the objective function to minimize the material handling cost or travel distance. The second problem seeks to determine the configuration of internal layout, such as the number and dimensions of aisles in the picking area. The common objective of this problem is to seek a "best" layout accounting for travel distance in most cases. [1]

Bassan *et al*. (1980) [17] introduced a layout design for a low-level picking system focusing on unit loads. They compare handling and layout costs of two parallel-aisle layouts. Rosenblatt and Roll [18] studied the effect of storage police to design warehouse layout. The authors also studied the effect of random demand and multiple service levels on layout and storage capability. Roodbergen [13] proposed a nonlinear programming approach. This approach considers average time spent on travel in terms of pickup location and number of picks per trip in order to determine aisle arrangement for random storage warehouses with objective of minimizing average travel distance. Caron *et al*. [19] and Le-Duc and De Koster [20] studied cube-per-order index (COI) -based storage assignment and class-based storage assignment also with an objective to minimize average travel distance. Peterson

[21] studied the effect of number of aisles as well as aisle dimension on total trip length by implementing simulation techniques.

2.4.2 Storage Location Assignment Problem

Five storage assignment methods have been introduced in previous studies: random storage [12] [22], closest open location storage [23], dedicated storage, COI storage [24], full turnover storage [25] [26] [27] [28], class based storage [29] [30] [31].

Random storage assigns SKU's with similar characteristics to random locations in the warehouse [8]. This method results in less allocation time, but higher space utilization [22]. This rule works well in an automated environment. In the closest-open-location storage method, the pickers choose the storage location within the warehouse. If shelves are full, the method defaults to the random storage policy [33]. In dedicated storage, since a specific location is reserved for a certain product, it's easier for the pickers to get familiar with the production location. The problem is that it causes waste of space when this product is out of stock. This method has the worst performance in term of space utilization comparing to all other storage methods. Full turnover storage policy is a kind of development of cube-per-order index (COI) rule. The COI of a product equals the total space requirement divided by total required trips to meet its demand per period. COI rules suggest locating the lowest COI item closest to the I/O point [12] [22] [23]. The full turnover storage method assigns products to locations based on their turnover ratio. The products with highest turnover ratios are allocated closest to the I/O point, while the products with the lowest turnover ratios are assigned in the back of the storage area. The biggest disadvantage of such method is that the demand changes frequently and layout reconfiguration is required every time demand changes. An information intensive system may be required in full turnover storage method to classify and allocate the products [24]. Class-based storage can be considered as the combination of several methods we have reviewed so far. The class is divided based on Pareto's method. That is, the classes with the fastest moving products contain 15% of the SKUs, but they contribute to the 85% of the system's turnover. Each class is assigned to a fixed location within the warehouse according to this Pareto rules.

Petersen and Schmenner [34] investigated the order picking efficiency of different storage policies and routing methods that consider order size and demand. The within-aisle method is 10–20% better in travel distance than other storage assignment policies. Petersen *et al.* [14] showed that full turnover storage has better performance than class-based storage in a manual order picking environment, but class-based storage with two to four classes was still recommended in practice. Chen *et al.* [35] introduced a dynamic operational method and a two-stage four-step heuristic to determine preliminary solutions, followed by a tabu search algorithm to find further improvements. Heskett first studied the COI storage policy in 1963 [24]; he focused on storage of frequently ordered items and smaller requirement of storage location closest to the I/O point. An integer linear programming (ILP) model is introduced to formulate this storage policy [24].

Frazelle and Sharp [15] pointed out that the COI policy can be improved especially for the case that multiple associated items are included in the same order. Yi-Fei Chuang [36] introduced a two-stage Clustering-Assignment Problem Model (CAPM) and a z-type picking path considering the associations among items. The CAPM improves results by more than 45% over the randomized method. Order Oriented Slotting (OOS) policy proposed by Mantel et al. [37] states that the SKUs need to be stored in the warehouse with a shortest total picking distance. Besides assigning the popular SKUs to the locations close to the I/O point, SKUs with a high association should be close to each other to minimize order picking travel distance.

A recent study by Diaz [38] indicated customer demand pattern, order clustering as well as physical restriction (i.e. weight or volume) are critical for improving operational performance of warehousing activities. A quadratic integer programming is used to generate the layout considering dynamic demand including throughput-to-storage ratios and order similarities. A simulation approach is applied to investigate the effect of implementing the generated layout in different zones classified by density. Weight is taken into consideration to ensure the heavy product is picked up first to avoid any physical damage of the product during order picking.

Literature that considers physical characters of products is not abundant. Most of the papers take similarity into consideration in the layout design and storage assignment phase to achieve the best objective value. But in practice, physical characters such as weight, area, and volume are one of the most important regulations. Also, in the published literature, the selected sample data set is relatively small due to the computation time of Storage Location Assignment problem phase. In this thesis, weight is used to divide unique products into classes, the improvement method is applied in each weight class to ensure meeting the regulation that heavy products need to be picked up first. For the storage location assignment problem, an exchange heuristic is implemented to deal with the large data set and produce a practical solution.

2.5 ARM Algorithm

Associations rules conceptually refer to sets of objects describing the relationships between items. ARM is an unsupervised mining technique, also known as affinity analysis or market basket analysis. ARM is widely used for targeted marketing, such as advertisement pushes, product recommendation and so on. By applying ARM, we can investigate whether two or more items are purchased together, or whether the purchase of one product increases the possibility of purchasing the other. The result from data mining analysis can provide suggestions for new layouts of a warehouse, when to push ads, when to issue coupons among offers. This thesis considers three metrics in ARM algorithms, support, confidence and lift [39], [40].

ARM algorithm is developed from an Apriori algorithm to identify groups of products that are ordered together, and find the frequency with which these item sets are ordered. The Apriori algorithm is efficient and relies on the downward closure lemma, which is used to discards sets that do not meet minimum support or minimum confidence conditions [39].

Apriori Pseudo-code [39]:

```
C: a candidate item set of size k $_k$
L: frequent item set of size k $_k$
T: database of transactions/trips
Apriori (T, ε)
L $_1$ ← {large 1-itemsets appear in more than ε transactions}
k←2
while ( L $_{k-1}$ ≄∅ )
    C $_k$ ← generate item sets from L $_{k-1}$
    for(transactions t∈ T)
        C ← subset (C $_k$, t) generate candidate transactions size k
        for (candidates c∈ C $_t$ )determines frequency of c-candidates
            count[c]←count[c] + 1
    L $_k$ ← { c : c ∈ C $_k$ ∧ count[ c ] ≥ ε } P r u n i n g
    k←k+1
return ∪$_k$L $_k$ union of sets of frequent items k=1, 2,... K
```

The result is an item set L_k, which consists of those item sets of sizes less than or equal to k that meet the minimum support required ε.

Chen *et al.* [4] introduced the method of creating order batching via ARM algorithms. Chen *et al.* [5] introduced an extension of this model, which models capacity constraints of the facility by using 0-1 integer programming. Chiang *et al.*[3] introduce two heuristics, modified class-based heuristic and association seed based heuristic, to solve storage location assignment problem considering relationship between products. These two heuristics assign correlated products which are frequently ordered together to the same aisle to maximize the association measure within the same aisle.

In the reviewed literature, the selected sample data set for scenarios is relatively small. It's rarely mentioned about the effect of combination of support and confidence in ARM. In this thesis, ARM algorithm is implemented to analyze larger data sets to achieve more rules via ARM. Combination of support and confidence is analyzed by applying design of experiment approach with specific data set to indicate the significant interaction of support and confidence.

2.6 Simulation Approach

Simulation is commonly used to visualize the order picking operations and analyze the outputs from different scenarios [38], [41], [42]. In the reviewed literatures, popular software such as Arena, Simio, and Witness are used to perform the simulation process. But for these packages, it's not time efficient to build large warehouse layout. In this thesis, the simulation model is built in AutoMod which is a package based on C language and focuses on logistic simulation and analysis. The model is generated by coding in a general way, which makes it easy to expand and modify.

III. METHODOLOGY

3.1 System Configuration.

The system under analysis is a DC that receives daily orders from 700 retail stores. The DC fills up these orders in the morning and replenishes the items sent by their suppliers at night. There are 70,000 SKU's. The DC has 52 aisles, each with 52 pickup locations, for a total of 2704 pickup locations. The distance between two adjacent pickup locations is 40 feet. Orders, each containing between 150 to 250 SKU's, are picked up by a human operator. The operator follows an S-shape route while picking up the orders from the picking slots. The operator rides a vehicle, which moves at 3 feet/sec. The pallet is formed by placing the items with more weight at the bottom to form the base of the pallet; items with less weight are placed on top of heavier items to preserve the integrity of the items. Fragile SKUs are placed on the top of the pallet. Figure 6 shows the layout under study.

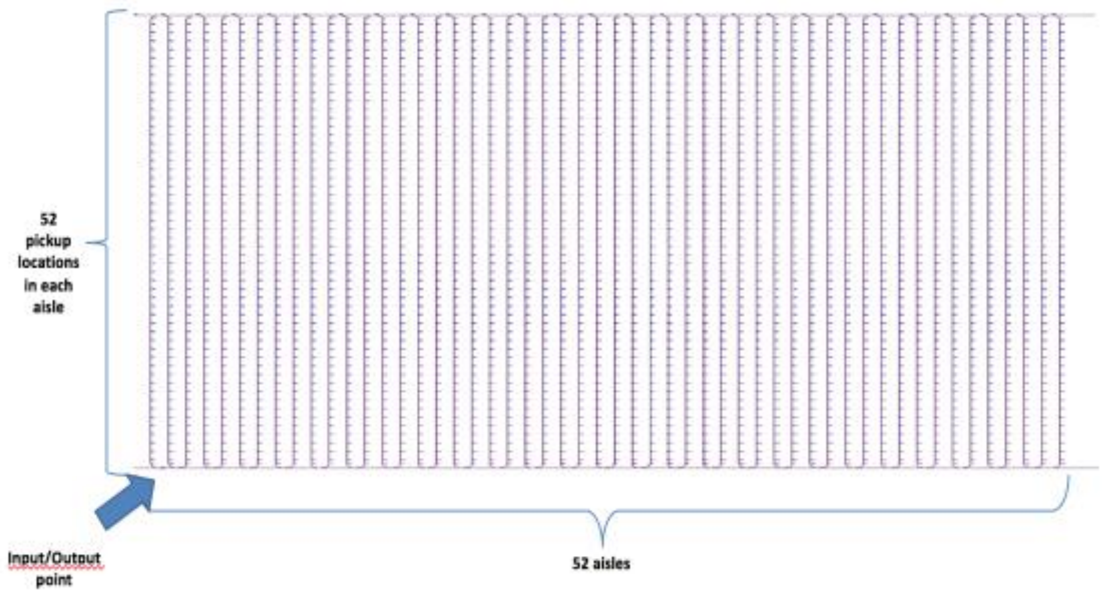


Figure 6 Layout assumption

3.2 Mathematical Formulation

In the Quadratic Assignment Problem (QAP) stated in (1)-(4). We consider number of SKUs and locations as *n*, the SKU *i* is assigned to location *j*, and each location *j* can only have one SKU *i* allocated.

Objective function of the problem is as follow to minimize the total joint measurement when SKU *i* is assigned to location *j*:

$$Minimize \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \frac{1}{2} S_{ij} D_{kl} X_{ik} X_{jl} + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij} \quad (1)$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \quad for\, j = 1,2, \dots, J \quad (2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad for\, i = 1,2, \dots, I \quad (3)$$

$$x_{ij} = (0,1) \quad \forall i, j \quad (4)$$

The decision variable is x_{ij} which is a binary variable with $x_{ij} = 1$ if SKU *i* is assigned to location *j* and $x_{ij} = 0$ if SKU *i* is not assigned to location *j*. The objective function represented in Equation (1) indicates minimizing the total joint measurement when SKU_i is assigned to $location_j$. S_{ij} is the dissimilarity of two SKUs which is obtained from ARM phase. D_{kl} is the distance between pick up locations. C_{ij} is the cost of assigning SKU *i* to location *j*. (2) is a constraint that ensures one SKU is assigned to one location. (3) is a constraint that ensures one location only contains one SKU. QAP is a NP-hard problem, as number of SKUs *n* increases, the computational time increases significantly. Therefore, a heuristic is considered to achieve a quick practical solution of storage assignment problem.

3.3 Solution Approach

Due to the computational complexity of QAP, it costs too much time to get the optimal solution which is against our objective of producing quick practical solution. A case study is conducted in next section to compare the computational time and performances between optimal and proposed heuristic. A five-stage approach is implemented in this thesis to solve the storage location assignment problem stated above. This approach is illustrated in Figure 7. The method divides unique SKUs into classes based on their weights (in pounds), and then ranks and sorts, from highest to lowest, the SKUs in each weight class based on order frequency. For example, a SKU is classified in class #2, and it is ordered the most frequently among all the SKUs within this class. This SKU should be allocated in the closest location to I/O point compared to other SKUs within this weight class. ARM algorithm is used to search correlated SKUs to create rules. Then, based on the association rules, the exchange heuristic reassigns the consequence SKUs to the locations after their antecedent SKUs. Simulation model is developed in AutoMod to visualize the order picking operation and to compare performances of different scenarios. These stages will be explained in detail below.

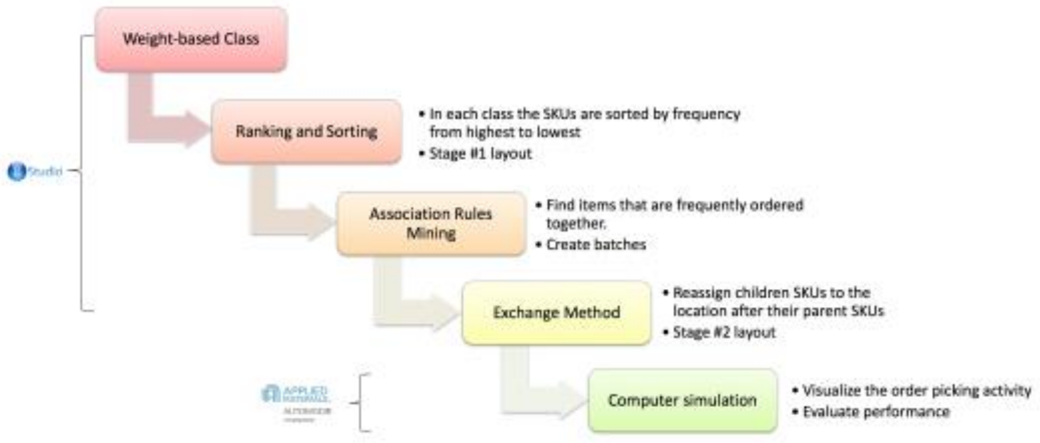

Weight-based Class
Ranking and Sorting
• In each class the SKUs are sorted by frequency from highest to lowest
• Stage #1 layout
Association Rules Mining
• Find items that are frequently ordered together.
• Create batches
Exchange Method
• Reassign children SKUs to the location after their parent SKUs
• Stage #2 layout
Computer simulation
• Visualize the order picking activity
• Evaluate performance

Figure 7 Methodology flow chart

3.3.1 Weight-based Class

Class-based storage strategy has positive impact on increasing order picking efficiency. In this thesis, the unique SKUs are divided into classes based on their weights (in pounds). The heaviest SKU in each class is allocated closest to entry point because, in practice, heavy SKUs need to be picked up first and put on the bottom of the pallet followed by lighter SKUs, which are placed on the top to avoid physical damages. This process is done in R Statistical Package by sorting the given weight of SKUs in the dataset from heaviest to lightest. The number of SKUs in each class may vary due to different data sample selection. The information needs to be determined as shown in Table 1 below.

Table 1 Weight-based classes

	Class #1 (Heavy)	**Class #2 (Medium)**	**Class #3 (Light)**
Weight(lbs)	>2/3 * Max{Weight i}	1/3 * Max{Weight i} to 2/3 * Max{Weight i}	<1/3 * Max{Weight i}
Number of SKUs	N1	N2	N3
Percentage	P1%	P2%	P3%

3.3.2 Ranking and Sorting

According to Diaz (2016) [38], the SKUs with the highest demand are visited most frequently, thus contributing to the travel volume for the order-picking operations. Placing these SKU’s closer to the I/O point helps decreasing the total travel distance. For this reason, the SKUs are ranked and sorted in each weight-based class by their frequency from highest to lowest. Due to consideration of both weight and frequency, sorting SKUs in a weight class based on frequency probably move a light, but frequently ordered SKU, to the location closer to entry point than a heavy but less frequently ordered SKU. Executing the ranking and sorting phase within each class reduces such risk because we consider the SKUs in each class are consistent in weight and make it acceptable to change their sequence. The result of this phase is Stage #1 Layout -an improved layout containing the storage locations of SKU's based on both weight and order frequency in each class. Since the pickup routing method is selected as S-Shape routing method, the layout can be considered as linear. See Figure 8.

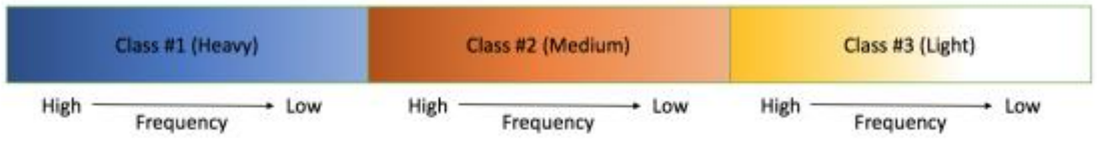


Figure 8 Stage #1 layout considering weight and frequency

3.3.3 ARM Algorithm

ARM generates rules of the form:

$$A \rightarrow B$$

The left hand (A) is usually called antecedent and the right hand (B) is called consequent. See Table 2 for examples of association rules. Different antecedents can be associated with the same consequent, and association rules are directional. That is, $A \rightarrow B$ and $B \rightarrow A$ are different rules. Figure 9 shows the graph of the rules. The color of arrows in the graph demonstrates the strength of the rules, a darker color means the rule is stronger.

Table 2 Examples of association rules

	LHS	RHS	support	confidence	lift
1	263639	31217	0.21	0.9	2.67
2	241863	111391	0.23	0.9	2.06
3	39091	179582	0.21	0.8	1.95
4	110142	111493	0.21	0.8	1.95
5	7217	45258	0.23	0.8	1.88

This thesis considers three metrics in ARM Algorithms: support, confidence and lift.

- Support indicates the joint probability of SKUs A and B are ordered together $P(A \cap B)$. $P(A \cap B)$ or sup (A and B) is equal to number of transactions that contain both item A and item B divided by total number of transactions.
- Confidence refers to the conditional probability that SKU B occurs given SKU A occurs, $P(B|A)$. That is, $P(B|A) = P(A \cap B)/ P(A)$ or $P(B|A) = supp(A \text{ and } B)/ supp(A)$.
- Lift is the strength of the rules. $Lift(A \rightarrow B)$ is calculated as $P(B|A)/P(B) = P(B \cap A)/P(A)P(B)$. If *lift* > 1, A and B are positively correlated, that is, if item A is ordered, it is more likely that item B is also order. If *lift* < 1, A and B are negatively correlated, which means, if a custom orders item A, it is more likely that the customer will NOT order item B. If *lift* = 1, item A and item B are independent items, that is, purchase of item A and purchase of item B are two independent events. Therefore, a high lift suggests the rule, if item A, then item B, may be a useful rule and a low lift suggests that the rule, if item A, then NOT item B, may be a useful rule.

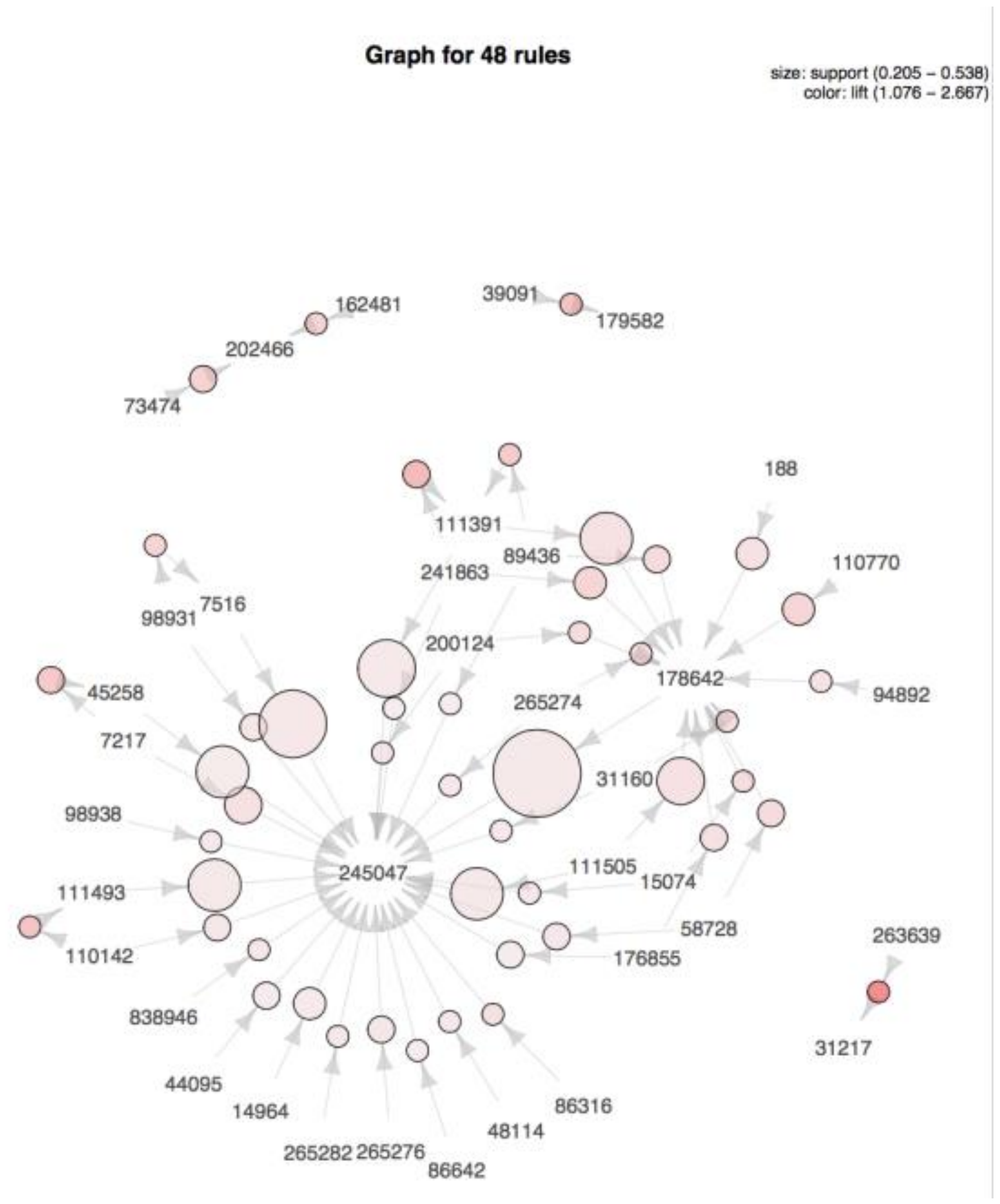


Figure 9 Example of ARM algorithm visualization

Association Rules are solved in this thesis using R. Figure 10 summarize the steps involved in the R calculations.

Step 1: Install and load required packages “arules” and “arulesViz”, if the package is installed correctly go to step 2, otherwise back to step 1

Step 2: Import dataset

Step 3: Reformat and transform dataset

Step 4: Duplicate check to obtain unique dataset, if the dataset is unique, go to step 5, otherwise go back to step 4.

Step 5: Define support and confidence level

Step 6: Apply data mining algorithm to produce association rules

Step 7: Inspect rules, if number of rules we expected, go to step 8, otherwise go back to step 5

Step 8: Sort rule and export

Step 9: Stop

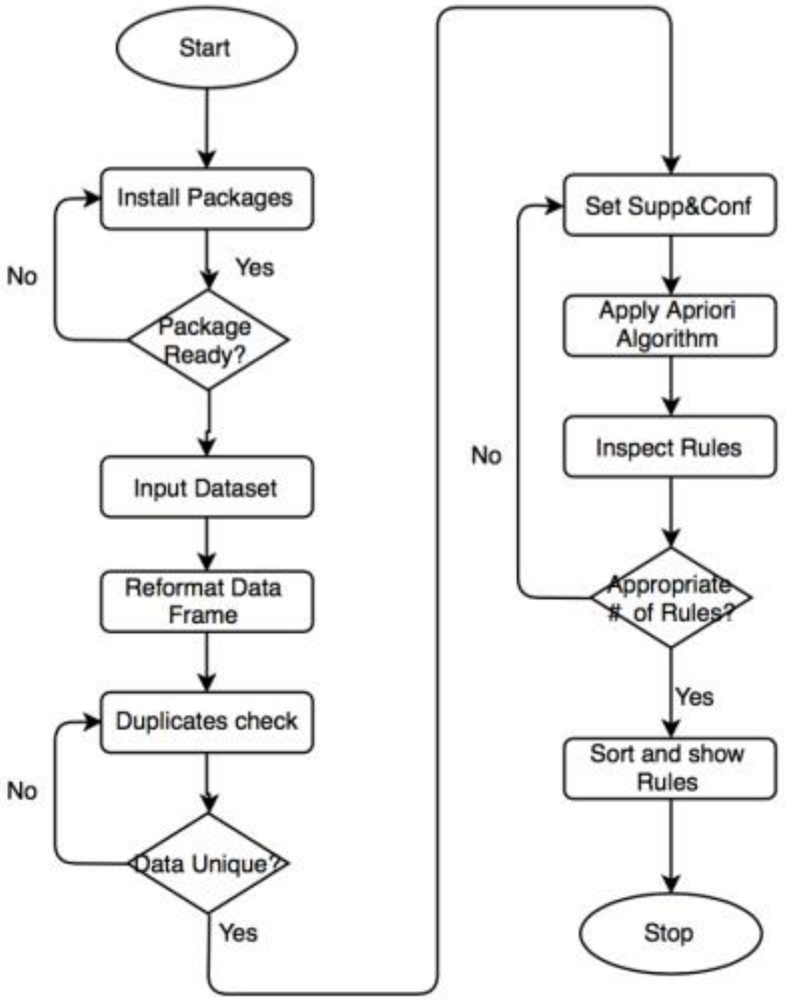


Figure 10 The work process of ARM in R package

3.3.4 Exchange Heuristic

Let A={a_1, a_2,…a_n} be a set of antecedents containing SKU's and $C=\{c_1,c_2,\ldots c_m\}$ be a set of consequents containing SKU's. The rules A→C are stored in a set of paired SKUs $R=\{(a_1,c_1),(a_2,c_2),\ldots(a_n, c_m)\}$ sorted by the support metric sup(A,C) in ascending order. A buffer set, with initial status of empty B={ϕ}, is generated to store the selected antecedents and consequents temporarily. The exchange heuristic, shown visually as a flow diagram in Figure 11, follows the steps stated below to produce new sequence of SKUs:

Step 1: Let i=1

Step 2: Select the *ith* element c_i in set C

Step 3: Check the corresponding element in set R, find the antecedent element a_i which is correlated with element c_i in R.

Step 4: Search a_i in set A

Step 5: Assign c_i from set C and add it to set B as b_i

Step 6: Insert b_i to set A after element a_i so that set $A = \{a_i, b_i, a_{i+1}, a_{i+2},\ldots,a_{n+1}\}$. Then empty set B

Step 7: Let $i=i+1$and go back to step 1

Step 8: After finishing the loop of m times, the set A includes $n+m$ elements so that $A = \{a_1, c_1, a_2, c_2, a_3, c_3,\ldots,a_n, c_m\}$

> Note: For special cases when different antecedents are correlated with the same consequents, the consequents choose the location of the antecedent that has the highest joint support. If a special case occurs, continue the heuristic algorithm with step 9, otherwise terminate the algorithm.

Step 9: unify set A, then set A contains the first occurrence of each duplicated element. For example, if A = {a_1, c_1, a_2, c_2, a_3, c_2, a_4, $c_3\ldots a_n$, c_3}, the unique A = {a_1, c_1, a_2, c_2, a_3, a_4, $c_3\ldots a_n$}

Step 10: Terminate algorithm

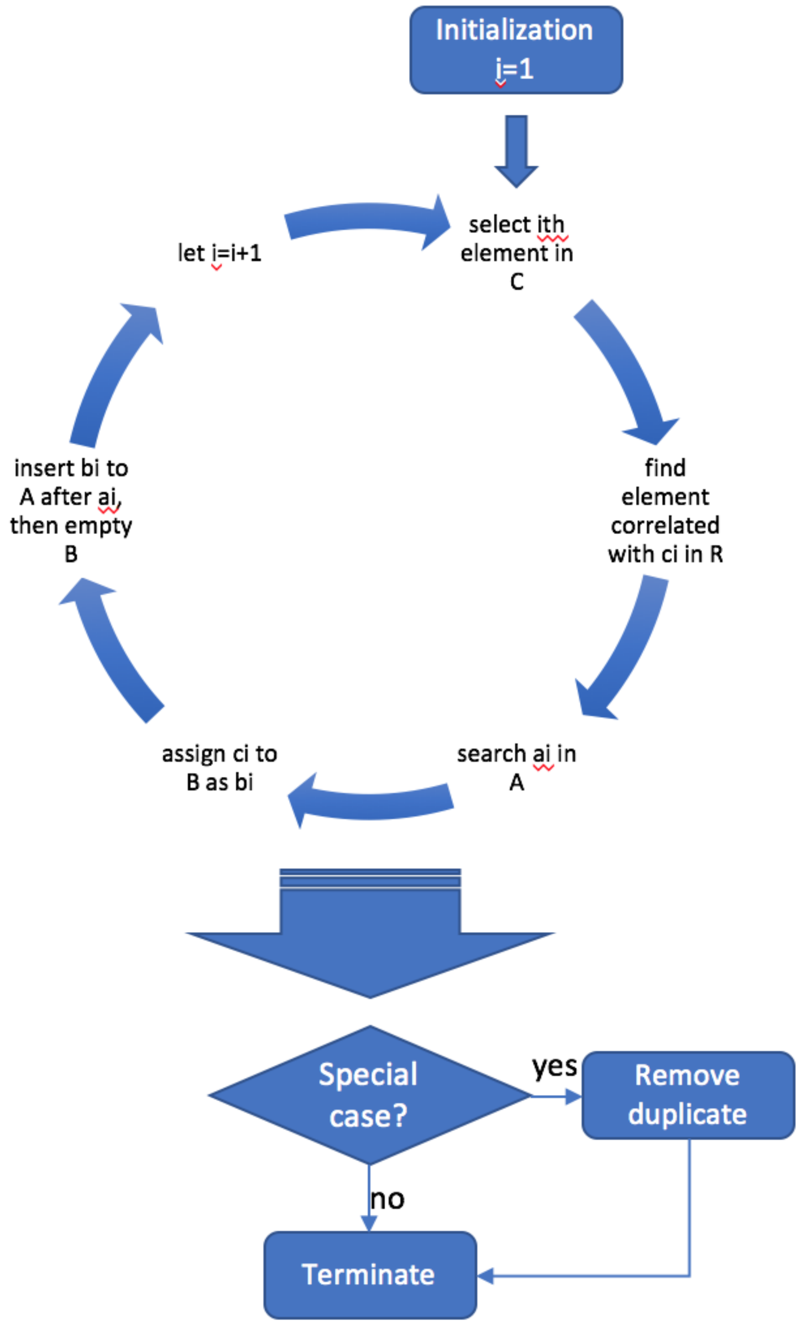


Figure 11 Illustration of the exchange heuristic

3.3.5 Computer Simulation

In this phase, a static simulation model of the system described in Section 3.1 was developed in AutoMod to test the output of different scenarios. The reason for choosing AutoMod instead of other software is that AutoMod has built-in subsystems to simulate logistic and warehousing operations. Furthermore, AutoMod is based on C language which is efficient to generate large layout with thousands of operation locations in a very general way to make the model convenient to modify and expand.

There are totally 1 process, 2 order lists, 5 attributes, 7 variables, 3 tables, 2 functions and 4 subroutines to control the logic of the model. Functions are used for initializing the orders and pickup locations, subroutines control the pickup operations, attributes and variables are used for identifying orders and locations while the tables are used for tabulating the results collected from simulation. See Table 3 for detail.

Table 3 Functions and parameters in simulation model

Category	Name	Type	Dimension	Description
Function	F_Order_Init	integer		Initialization of orders
	F_Location_Init	integer		Initialization of locations
Subroutine	SPickOrder			Subroutine of order picking process
	SReceiveOrder			Subroutine of order receiving process
	SDeliverOrder			Subroutine of order delivery process
	SParking			Subroutine of parking process
Process	PMasterLoad			Main process
Queue	QInitial	integer	1	Initial queue at entry point
Order list	OLAssignment	integer	1	List of assignments to be picked up
	OLVehicle	integer	1	List of available vehicles
Attribute	Attr_Order_ID	integer	1	Order ID
	Attr_Time	time	1	Counted time for waiting time calculation
	A_index	integer	1	Index of vehicle
	A_Home	location	1	Entry point
	Attr_Time2	time	1	Counted time for picking up time calculation
Variable	V_Location	location	2	Pickup locations
	V_Number_Item_Per_Ord	integer	2	Number of SKUs included in each order
	V_Number_Done	integer	1	Number of orders done
	V_Order	integer	3	Identifier of each SKU in each order
	V_Order_ctr	integer	1	Counter for order ID
	V_Count	integer	1	Counter for vehicle index
	Vi	integer	1	Counter for SKUs in an order
Table	T_PickTime	time	1	Table of pickup time
	T_WaitTime	time	1	Table of waiting time
	T_Throughput	integer	1	Table of throughput

Two scenarios are tested via simulation to compare the order-pickup time for different storage location assignments produced by ARM algorithm.

IV. EXPERIMENTS AND RESULTS

In this chapter, case studies are conducted to show the performance of the proposed five-step methodology described in Chapter 3.

4.1 Data Description

Our raw data set includes daily transactions from retail stores as provided by a local distribution center in Texas. The raw data set includes 624 different orders, which accounts for a total of 62,887 SKUs. The dataset also included related information such as the order number, store number, location of SKUs, SKU number, SKU description, quantity, weight, and volume, among other main fields. Particular emphasis is placed in the selected fields such as the order number, SKU number, location, weight and SKU description.

The dataset was partitioned to test for repeatability of the proposed algorithms while managing their computational time. For this reason, two partitions were obtained from the dataset. SKU's were selected at random. The first partition (referred to as Scenario 1) included 39 orders with a total amount of 8141 SKUs. The number of unique SKUs is 2638. The second partition (referred to as Scenario 2) included 39 orders with 8122 SKUs. The number of unique SKUs is 2721.

4.2 Effect of Support and Confidence

Support and confidence are defined in R to produce different number of association rules. In this section, different scenarios are tested exclusively to identify appropriate values of support and confidence required to obtain an adequate number of rules from ARM algorithm described in Section 3.3.3. The code was developed in a R package named "arules" version 1.5-2. [43]. It was run on a computer type of MacBook Pro with an Intel

2.5 GHz Core i7 processor and 16 GB 1600 MHz DDR3 memory. The program ran for 15 seconds. The code is shown in Appendix 3.

4.2.1 Design of Experiments

a. Two factors are taken in to consideration in this phase, support and confidence.
    - Support has 5 levels from 0.1 to 0.3 with increments of 0.05.
    - Confidence has 5 levels from 0.5 to 0.9 with an interval of 0.1.

b. Response
    - The response is the number of association rules generated from ARM in R. For example, number of rules $A \rightarrow B$ where A is antecedent and B is consequent.

c. Run Design
    - 2 factors, each with 5 levels, are taken into consideration. The total number of combinations is 25.

d. Configurations of DOE
    - Factors: 2
    - Replicate: 2; one with the partition labeled as Scenario 1 and the second one with the partition labeled as Scenario 2.
    - Total runs: 50
    - Total blocks: 2
    - Number of levels: 5, 5

See Table 4 and Figure 12 for all the combinations of factors.

Table 4 Design table of two runs in DOE

Support	**Confidence**	**Response of Run1 (Scenario 1)**	**Response of Run2 (Scenario 2)**
0.1	0.5	11965	11244
0.1	0.6	6918	6280
0.1	0.7	4244	3618
0.1	0.8	3246	2750
0.1	0.9	1134	843
0.15	0.5	2651	2177
0.15	0.6	1832	1586
0.15	0.7	1068	913
0.15	0.8	585	448
0.15	0.9	188	125
0.2	0.5	708	525
0.2	0.6	511	376
0.2	0.7	335	238
0.2	0.8	215	159
0.2	0.9	62	50
0.25	0.5	218	136
0.25	0.6	182	102
0.25	0.7	115	79
0.25	0.8	64	39
0.25	0.9	23	19
0.3	0.5	93	38
0.3	0.6	73	27
0.3	0.7	58	22
0.3	0.8	35	14
0.3	0.9	8	5

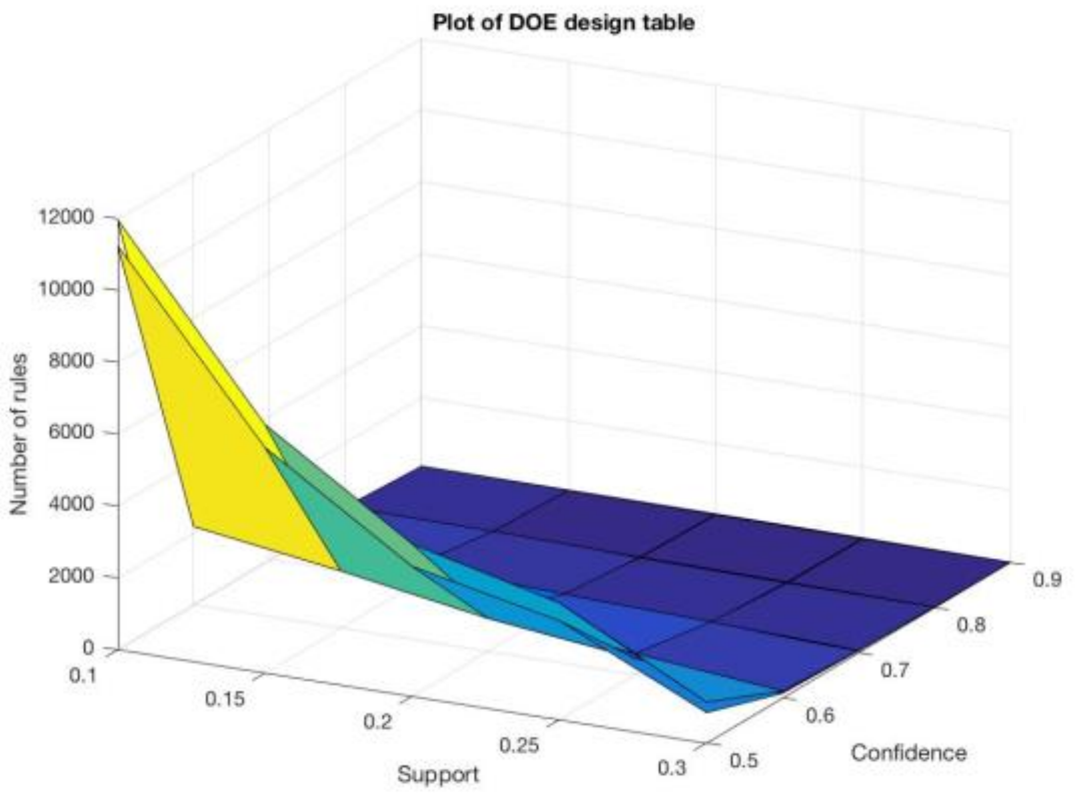
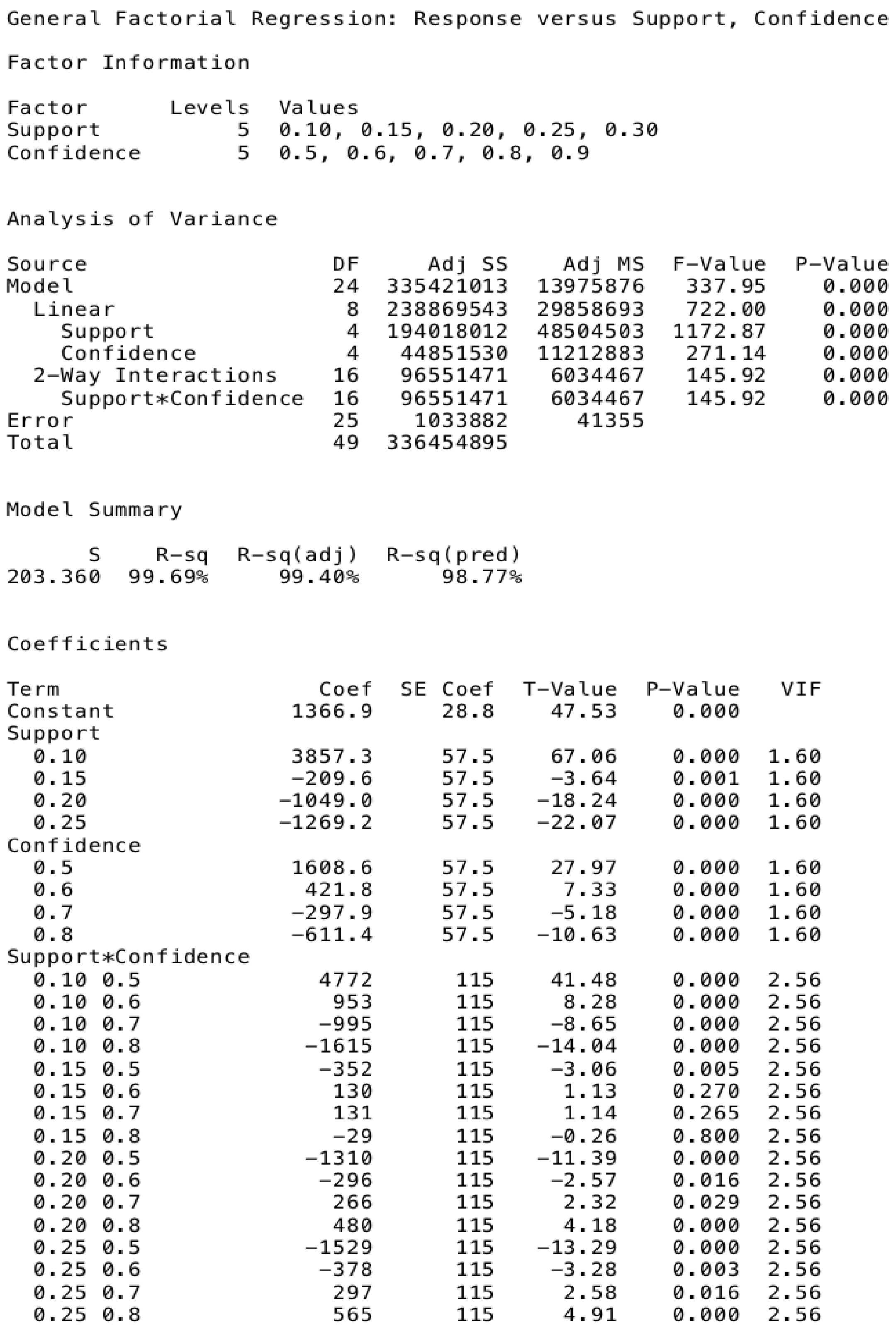


Figure 12 Plot of DOE design table

4.2.2 Output Analysis

Results from different scenarios show that as the support and confidence increase, the number of association rules as well as computation time decrease. As support and confidence increase, the selection standard becomes strict, which means fewer sets are taken into consideration and the SKUs involved in the transactions are highly correlated.

By analyzing the factorial regression model, p-value less than 0.05 indicates that 8 main factors and 13 interactions have statistically significant effects on the response. See Figure 13, 14 for details.



```
General Factorial Regression: Response versus Support, Confidence

Factor Information

Factor      Levels  Values
Support          5  0.10, 0.15, 0.20, 0.25, 0.30
Confidence       5  0.5, 0.6, 0.7, 0.8, 0.9


Analysis of Variance

Source                  DF     Adj SS    Adj MS  F-Value  P-Value
Model                   24  335421013  13975876   337.95    0.000
  Linear                 8  238869543  29858693   722.00    0.000
    Support              4  194018012  48504503  1172.87    0.000
    Confidence           4   44851530  11212883   271.14    0.000
  2-Way Interactions    16   96551471   6034467   145.92    0.000
    Support*Confidence  16   96551471   6034467   145.92    0.000
Error                   25    1033882     41355
Total                   49  336454895


Model Summary

      S    R-sq  R-sq(adj)  R-sq(pred)
203.360  99.69%     99.40%      98.77%


Coefficients

Term                    Coef  SE Coef  T-Value  P-Value   VIF
Constant              1366.9     28.8    47.53    0.000
Support
  0.10                3857.3     57.5    67.06    0.000  1.60
  0.15                -209.6     57.5    -3.64    0.001  1.60
  0.20               -1049.0     57.5   -18.24    0.000  1.60
  0.25               -1269.2     57.5   -22.07    0.000  1.60
Confidence
  0.5                 1608.6     57.5    27.97    0.000  1.60
  0.6                  421.8     57.5     7.33    0.000  1.60
  0.7                 -297.9     57.5    -5.18    0.000  1.60
  0.8                 -611.4     57.5   -10.63    0.000  1.60
Support*Confidence
  0.10 0.5              4772      115    41.48    0.000  2.56
  0.10 0.6               953      115     8.28    0.000  2.56
  0.10 0.7              -995      115    -8.65    0.000  2.56
  0.10 0.8             -1615      115   -14.04    0.000  2.56
  0.15 0.5              -352      115    -3.06    0.005  2.56
  0.15 0.6               130      115     1.13    0.270  2.56
  0.15 0.7               131      115     1.14    0.265  2.56
  0.15 0.8               -29      115    -0.26    0.800  2.56
  0.20 0.5             -1310      115   -11.39    0.000  2.56
  0.20 0.6              -296      115    -2.57    0.016  2.56
  0.20 0.7               266      115     2.32    0.029  2.56
  0.20 0.8               480      115     4.18    0.000  2.56
  0.25 0.5             -1529      115   -13.29    0.000  2.56
  0.25 0.6              -378      115    -3.28    0.003  2.56
  0.25 0.7               297      115     2.58    0.016  2.56
  0.25 0.8               565      115     4.91    0.000  2.56
```

Figure 13 Output of design of experiment

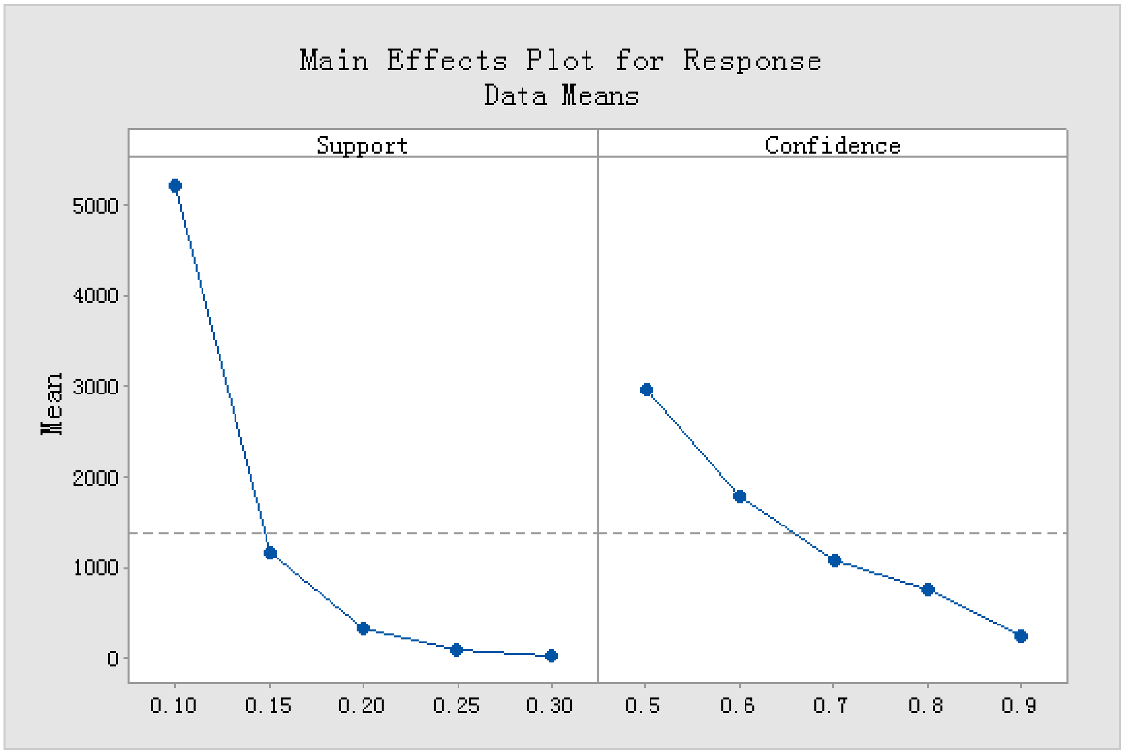


Figure 14 Main effects plot for response

By balancing the number of rules and computational time, support range from 0.2 to 0.25 and confidence range from 0.5-0.8 is appropriate in our case. In this thesis, preferred support and confidence in ARM in R are selected as 0.2 and 0.8 to produce association rules.

4.3 Comparison between Optimal and Heuristic Solution

In this subsection, the comparison between optimal and heuristic solution is conducted to prove that the heuristic is implementable in practice.

As is mentioned in Section 3, it is difficult to solve quadratic assignment problem and produce the optimal solution due to the complexity and computational time of the problem. The computational time increases significantly when number of SKUs increases. In this thesis, computational time and performance between optimal (in AMPL) and proposed heuristic results are compared. See Table 5 and 6. AMPL code for problem formulation is shown in Appendix 1.

Table 5 Comparison of computational time

	In AMPL via solver CPLEX	**Heuristic in R**
Number of SKUs	**Computational time (in second)**	**Computational time (in second)**
10	2	<30
15	231	<30
20	10310	<30
~8000	N/A	~300

Table 6 Comparison of order picking time

Order picking time comparison

	Optimal	Heuristic
Number of SKUs	10	10
Number of orders	14	14
SKU per order	1~5	1~5
Order picking time (in second)	237.244	254.522

The difference in term of order picking time between optimal and heuristic solution is 17.278 seconds. The optimal solution is 6.788% better than the heuristic solution in this specific case.

By considering the computational time and performance, the heuristic is implementable to achieve a good practical solution with large data set in a much shorter time when compare to the optimal solution.

4.4 Numerical Analysis

4.4.1 System Configuration

a. Data size

For Scenario 1, the final selected sample data set include 39 orders with 8141 SKUs. Class #1 includes 802 unique SKUs, Class #2 includes 756 unique SKUs and Class #3 includes 1080 unique SKUs.

For Scenario 2, the final selected sample data set include 39 orders with 8122 SKUs. Class #1 includes 805 unique SKUs, Class #2 includes 902 unique SKUs and Class #3 includes 1014 unique SKUs.

b. Metric

Support: 0.2

Confidence: 0.8

Batch size: 2

For Exchange Heuristic phase, the configuration is shown as follow:

a. Data size

Scenario 1 has 76 batches of size = 2.

Scenario 2 has 59 batches of size = 2.

b. Initial status

Before applying the method, all the rules obtained within each weight-based class are sorted by lift from highest to lowest which means the strongest association rule has higher priority to be allocated.

The configuration of Computer Simulation phase is shown as follow:

a. Data size

For scenario 1, 39 orders with 150-250 SKUs in each including totally 8141 SKUs are used to build simulation model; 2638 unique SKUs are used to create facility layout. For scenario 2, 39 orders with 150-250 SKUs in each including totally 8122 SKUs are used to build simulation model; 2721 unique SKUs are used to create facility layout. R code for data selection is shown in Appendix 2.

b. Simulation settings

Two scenarios have the same assumptions for layout, vehicle, pickup and routing method *etc*. The differences are the SKU locations in the layout and the association rules that are considered. See Table 7. The assumptions are made considering both the complexity of simulation model as well as the real behaviors in real world. For example, we fixed the conditions, such as the layout, pick up and routing method to reduce the complexity in order to purely compare the performances of two layouts. The assumptions of vehicle and distance are based on the study of dataset and the pickup behaviors.

Table 7 Configurations of two scenarios

	Scenario 1	**Scenario 2**
Number of aisles	52	52
Pickup locations in each aisle	52	52
Distance between adjacent locations (feet)	40	40
Number of vehicles	1	1
Velocity of vehicles (feet/sec)	3	3
Pickup Method	Picker-to-part	Picker-to-part
Routing Method	S-shape	S-shape
Number of order per trip	1	1
Support	0.2	0.2
Confidence	0.8	0.8
Number of orders	39	39
Total SKUs	8141	8122
Total Unique SKUs	2638	2711
Unique SKUs in Class 1	802(30.40%)	805(29.58%)
Unique SKUs in Class 2	756(28.66%)	902(33.15%)
Unique SKUs in Class 3	1080(40.94%)	1014(37.27%)
Number or rules	76	59
Comparisons	Random VS Stage #1 VS Stage #2	Random VS Stage #1 VS Stage #2

c. Run control

The simulation model runs for 1 replication since the model is deterministic. That is, all the controllable factors such as vehicle speed, picking up time, waiting time, routing selection are fixed. The run time of model vary from one scenario to another, See Table 8. The simulation time indicates the run time of the deterministic simulation model under each condition.

Table 8 Simulation time (in second)

	Scenario 1			Scenario 2		
	Random Layout	**Stage #1 Layout**	**Stage #2 Layout**	**Random Layout**	**Stage #1 Layout**	**Stage #2 Layout**
Replicates	1	1	1	1		1
Simulation Time	12,079,744	1,568,458	1,410,042	11,773,986	1,635,642	1,376,351

4.4.2 Weight-based Class

Both scenarios, Scenario 1 and Scenario 2, were divided into three weight classes. According to Peterson *et al*. [29], 2-4 classes are suggested in practice since it is easier to implement and required less to administer. For Scenario 1, 2638 unique SKUs are divided into three classes based on their weights. Class #1 accounts for 30.40% of all the SKUs in sample data set containing 802 SKUs with weight greater than 9 pounds. Class #2 includes 756 SKUs with weight between 5.28 and 9 pounds, which is 28.66% of all the sample SKUs. Class #3 has 1080 SKUs with weight less than 5.28 pounds. This class includes most of the SKUs which is 40.94% of all the SKUs in the selected sample. See Table 9.

For Scenario 2, 2721 unique SKUs are divided into three classes based on their weights. Class #1 accounts for 29.58% of all the SKUs in sample data set containing 805 SKUs with weight greater than 8.85 pounds. Class #2 includes 902 SKUs with weight between 5 and 8.85 pounds which is 33.15% of all the sample SKUs. Class #3 has 1014 SKUs with weight less than 5 pounds. This class includes most of the SKUs which is 37.27% of all the SKUs in selected sample. See Table 10.

Table 9 Weight-based classes for scenario 1

	Class #1(Heavy)	**Class #2(Medium)**	**Class #3(Light)**
Weight(lbs)	>9	5.28-9	<5.28
Number of SKUs	*802*	*756*	*1080*
Percentage	30.40%	28.66%	40.94%

Table 10 Weight-based classes for scenario 2

	Class #1(Heavy)	**Class #2(Medium)**	**Class #3(Light)**
Weight(lbs)	>8.85	5-8.85	<5
Number of SKUs	*805*	*902*	*1014*
Percentage	29.58%	33.15%	37.27%

4.4.3 Ranking and Sorting

For each weight-based class, the Rank and Sortation method is implemented separately and then combine again to achieve Stage #1 layout.

Scenario 1: Class #1 includes 802 unique SKUs with frequency vary from 23 to 1. Class #2 includes 756 unique SKUs with frequency vary from 16 to 1. Class #3 includes 1080 unique SKUs with frequency vary from 25 to 1. See Figure 15.

Scenario 2: Class #1 includes 805 unique SKUs with frequency vary from 20 to 1. Class #2 includes 902 unique SKUs with frequency vary from 25 to 1. Class #3 includes 1014 unique SKUs with frequency vary from 23 to 1. See Figure 16.

The rank and sortation phase is done in R by implemented functions in 'dplyr' and 'aruels' packages. The computation time of ARM in each class is about 15 seconds on a computer with i7 processor and 16GB memory. The code is shown in Appendix 4.

The output of this phase is Stage #1 layout which consider both weight and frequency of SKUs.

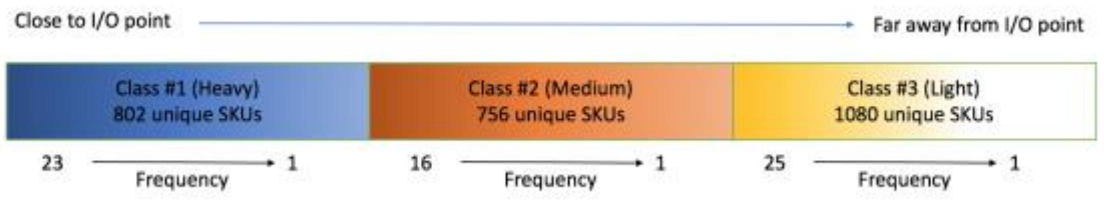


Figure 15 Stage #1 layout in scenario 1

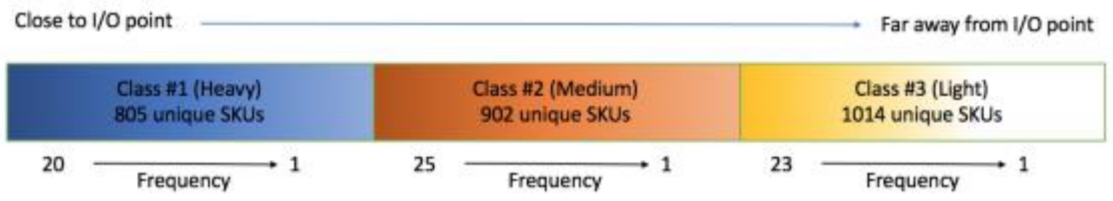


Figure 16 Stage #1 layout in scenario 2

4.4.4 ARM Algorithm

ARM is conducted in three weight-based classes to create batches of correlated SKUs.

Scenario 1: By implementing ARM algorithm in R, 10 association rules are obtained in Class #1, 48 rules are achieved in Class #2 while 18 rules are included in Class #3. Class #2 is the most correlation effective since it accounts for 28.66% of all the unique SKUs but contributes 63.16% of the correlated pairs. See Table 11,12,13.

Scenario 2: 14 association rules are obtained in Class #1, 32 rules are achieved in Class #2 while 13 rules are included in Class #3. See Table 14, 15, 16.

Table 11 Association rules of class #1 sorted by lift in scenario 1

	rules	support	confidence	lift
3	{189145} => {90991}	0.2051	0.8889	2.4762
6	{271586} => {90991}	0.2051	0.8000	2.2286
9	{143789} => {90991}	0.2051	0.8000	2.2286
4	{10829} => {103427}	0.2308	0.9000	1.4040
1	{117091} => {103427}	0.2051	0.8889	1.3867
2	{113849} => {103427}	0.2051	0.8889	1.3867
10	{83626} => {103427}	0.2821	0.8462	1.3200
5	{249707} => {103427}	0.2308	0.8182	1.2764
8	{241880} => {103427}	0.2308	0.8182	1.2764
7	{271586} => {103427}	0.2051	0.8000	1.2480

Table 12 Association rules of class #2 sorted by lift in scenario 1

	rules	support	confidence	lift
3	{263639} => {31217}	0.2051	0.8889	2.6667
29	{241863} => {111391}	0.2308	0.9000	2.0647
5	{39091} => {179582}	0.2051	0.8000	1.9500
18	{110142} => {111493}	0.2051	0.8000	1.9500
35	{7217} => {45258}	0.2308	0.8182	1.8770
32	{89436} => {111391}	0.2051	0.8000	1.8353
17	{73474} => {202466}	0.2308	0.8182	1.6794
10	{162481} => {202466}	0.2051	0.8000	1.6421
14	{98931} => {7516}	0.2051	0.8000	1.6421
28	{110770} => {178642}	0.2564	1.0000	1.6250
30	{241863} => {178642}	0.2564	1.0000	1.6250
25	{58728} => {178642}	0.2308	0.9000	1.4625
33	{89436} => {178642}	0.2308	0.9000	1.4625
6	{15074} => {178642}	0.2051	0.8889	1.4444
8	{200124} => {178642}	0.2051	0.8889	1.4444
12	{265274} => {178642}	0.2051	0.8889	1.4444
20	{31160} => {178642}	0.2051	0.8889	1.4444
40	{188} => {178642}	0.2564	0.8333	1.3542
1	{86316} => {245047}	0.2051	1.0000	1.3448
36	{7217} => {245047}	0.2821	1.0000	1.3448
45	{111391} => {178642}	0.3590	0.8235	1.3382
37	{176855} => {178642}	0.2308	0.8182	1.3295
42	{111505} => {178642}	0.3333	0.8125	1.3203
24	{94892} => {178642}	0.2051	0.8000	1.3000
15	{98931} => {245047}	0.2308	0.9000	1.2103
19	{110142} => {245047}	0.2308	0.9000	1.2103
23	{265276} => {245047}	0.2308	0.9000	1.2103
26	{58728} => {245047}	0.2308	0.9000	1.2103
47	{7516} => {245047}	0.4359	0.8947	1.2033
2	{838946} => {245047}	0.2051	0.8889	1.1954
4	{98938} => {245047}	0.2051	0.8889	1.1954
7	{15074} => {245047}	0.2051	0.8889	1.1954
9	{200124} => {245047}	0.2051	0.8889	1.1954
11	{265282} => {245047}	0.2051	0.8889	1.1954
13	{265274} => {245047}	0.2051	0.8889	1.1954
16	{48114} => {245047}	0.2051	0.8889	1.1954
21	{31160} => {245047}	0.2051	0.8889	1.1954
46	{111391} => {245047}	0.3846	0.8824	1.1866
41	{111493} => {245047}	0.3590	0.8750	1.1767
43	{111505} => {245047}	0.3590	0.8750	1.1767

	rules	support	confidence	lift
48	{178642} => {245047}	0.5385	0.8750	1.1767
39	{14964} => {245047}	0.2564	0.8333	1.1207
44	{45258} => {245047}	0.3590	0.8235	1.1075
27	{44095} => {245047}	0.2308	0.8182	1.1003
38	{176855} => {245047}	0.2308	0.8182	1.1003
22	{86642} => {245047}	0.2051	0.8000	1.0759
31	{241863} => {245047}	0.2051	0.8000	1.0759
34	{89436} => {245047}	0.2051	0.8000	1.0759

Table 13 Association rules of class #3 sorted by lift in scenario 1

	rules	support	confidence	lift
1	{52616} => {87168}	0.2051	1.0000	3.0000
7	{770099} => {630434}	0.2051	0.8889	2.8889
4	{77786} => {167155}	0.2051	1.0000	2.7857
13	{920785} => {630434}	0.2564	0.8333	2.7083
14	{630434} => {920785}	0.2564	0.8333	2.7083
8	{770099} => {167155}	0.2051	0.8889	2.4762
5	{24982} => {265030}	0.2308	0.9000	2.3400
15	{920785} => {167155}	0.2564	0.8333	2.3214
16	{630434} => {167155}	0.2564	0.8333	2.3214
9	{83551} => {265031}	0.2051	0.8889	2.3111
12	{119144} => {167155}	0.2308	0.8182	2.2792
17	{265031} => {117791}	0.3333	0.8667	2.1125
18	{117791} => {265031}	0.3333	0.8125	2.1125
10	{73329} => {3305}	0.2051	0.8000	2.0800
6	{24982} => {117791}	0.2051	0.8000	1.9500
11	{87168} => {73583}	0.3077	0.9231	1.5652
2	{176438} => {73583}	0.2051	0.8889	1.5072
3	{87167} => {73583}	0.2051	0.8000	1.3565

Table 14 Association rules of class #1 sorted by lift in scenario 2

	rules	**support**	**confidence**	**lift**
5	{73328} => {73325}	0.231	1.000	3.000
10	{84155} => {630434}	0.205	0.800	2.836
12	{630434} => {920785}	0.231	0.818	2.455
13	{460931} => {3305}	0.231	0.818	2.455
14	{170743} => {920785}	0.231	0.818	2.455
9	{269308} => {920785}	0.205	0.800	2.400
11	{84155} => {920785}	0.205	0.800	2.400
7	{41866} => {181846}	0.282	0.846	2.357
8	{41866} => {62134}	0.282	0.846	1.941
1	{62054} => {62134}	0.205	0.800	1.835
6	{87167} => {73583}	0.231	0.818	1.595
2	{62054} => {73583}	0.205	0.800	1.560
3	{265215} => {73583}	0.205	0.800	1.560
4	{87168} => {73583}	0.205	0.800	1.560

Table 15 Association rules of class #2 sorted by lift in scenario 2

	rules	support	confidence	lift
20	{110770} => {110142}	0.2051	0.8000	2.6000
26	{110142} => {111391}	0.2564	0.8333	2.5000
11	{87376} => {202466}	0.2564	0.9091	2.0856
14	{109567} => {179582}	0.2564	0.9091	2.0856
6	{152603} => {179582}	0.2308	0.9000	2.0647
7	{152603} => {202466}	0.2308	0.9000	2.0647
1	{111505} => {178642}	0.2051	1.0000	2.0526
4	{92733} => {179582}	0.2051	0.8889	2.0392
24	{269305} => {202466}	0.2821	0.8462	1.9412
15	{111493} => {178642}	0.2051	0.8000	1.6421
21	{110770} => {178642}	0.2051	0.8000	1.6421
2	{111505} => {245047}	0.2051	1.0000	1.5600
5	{265276} => {245047}	0.2308	1.0000	1.5600
9	{265274} => {245047}	0.2308	1.0000	1.5600
30	{7217} => {245047}	0.3333	0.9286	1.4486
31	{7516} => {245047}	0.3333	0.9286	1.4486
28	{111391} => {245047}	0.3077	0.9231	1.4400
25	{45258} => {245047}	0.2821	0.9167	1.4300
27	{110142} => {245047}	0.2821	0.9167	1.4300
16	{111493} => {245047}	0.2308	0.9000	1.4040
17	{444588} => {245047}	0.2308	0.9000	1.4040
19	{58728} => {245047}	0.2308	0.9000	1.4040
3	{87567} => {245047}	0.2051	0.8889	1.3867
8	{200124} => {245047}	0.2051	0.8889	1.3867
12	{89436} => {245047}	0.2051	0.8889	1.3867
13	{241863} => {245047}	0.2051	0.8889	1.3867
29	{14964} => {245047}	0.2821	0.8462	1.3200
32	{178642} => {245047}	0.4103	0.8421	1.3137
23	{241601} => {245047}	0.2308	0.8182	1.2764
10	{183555} => {245047}	0.2051	0.8000	1.2480
18	{109958} => {245047}	0.2051	0.8000	1.2480
22	{110770} => {245047}	0.2051	0.8000	1.2480

Table 16 Association rules of class #3 sorted by lift in scenario 2

	rules	support	confidence	lift
7	{183555} => {63831}	0.205	0.800	2.836
12	{19163} => {75638}	0.231	0.818	1.994
2	{113849} => {75638}	0.205	0.800	1.950
9	{91999} => {75638}	0.205	0.800	1.950
1	{441279} => {103427}	0.205	1.000	1.696
13	{19163} => {103427}	0.282	1.000	1.696
3	{113849} => {103427}	0.231	0.900	1.526
5	{10829} => {103427}	0.231	0.900	1.526
4	{246717} => {103427}	0.205	0.889	1.507
11	{63831} => {103427}	0.231	0.818	1.387
6	{86241} => {103427}	0.205	0.800	1.357
8	{183555} => {103427}	0.205	0.800	1.357
10	{91999} => {103427}	0.205	0.800	1.357

Figures 17- 22 show the plot of rules in each weight class in both scenarios. In these figures, size of circles stands for the support of rules, large size means higher support; the color shows the lift of the rules, darker color stands for higher lift. The arrow starts from the antecedent pointing to the consequents. Different SKUs can be correlated with the same SKU.

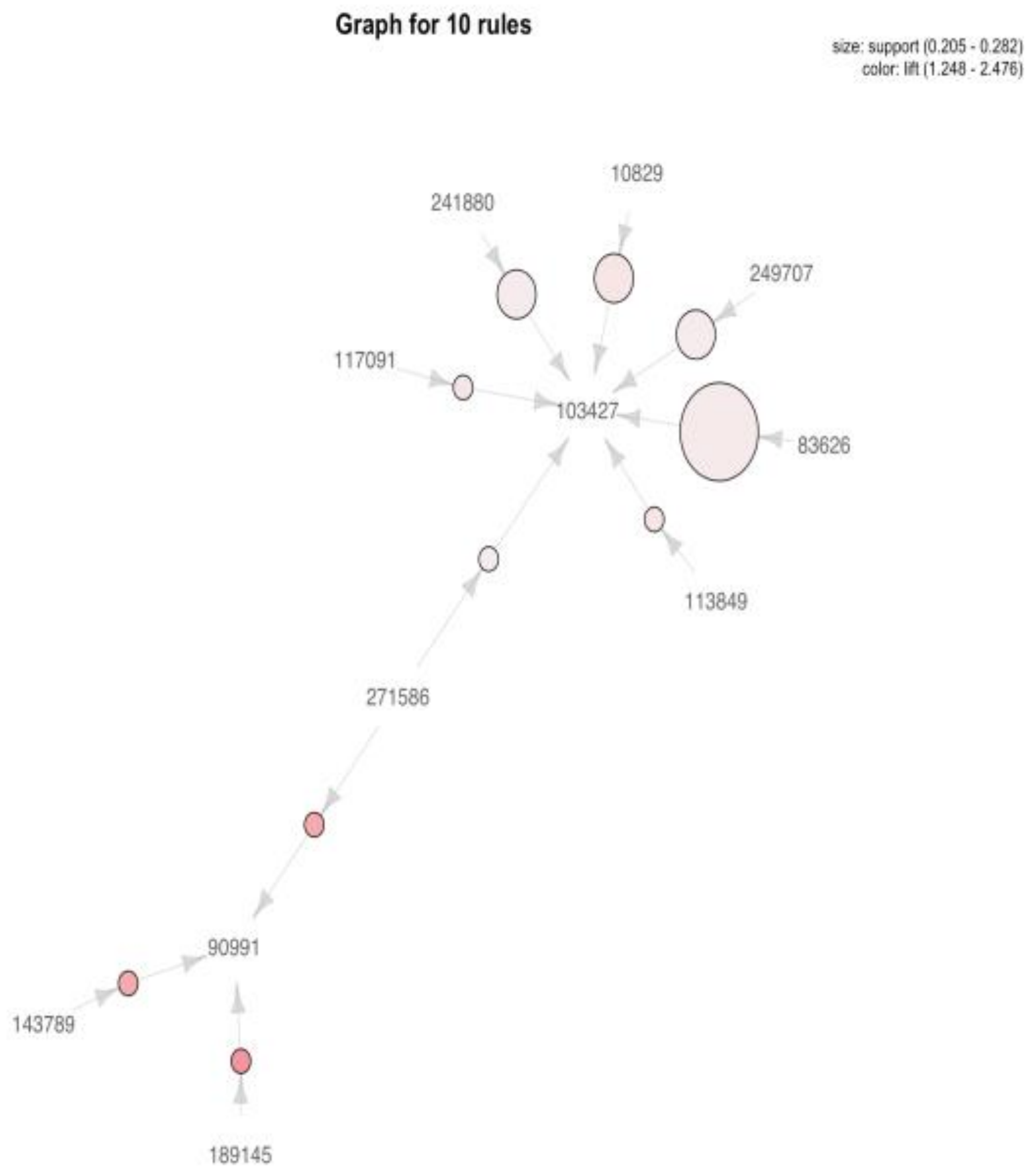


Figure 17 Rules plot of weight class #1 in scenario 1

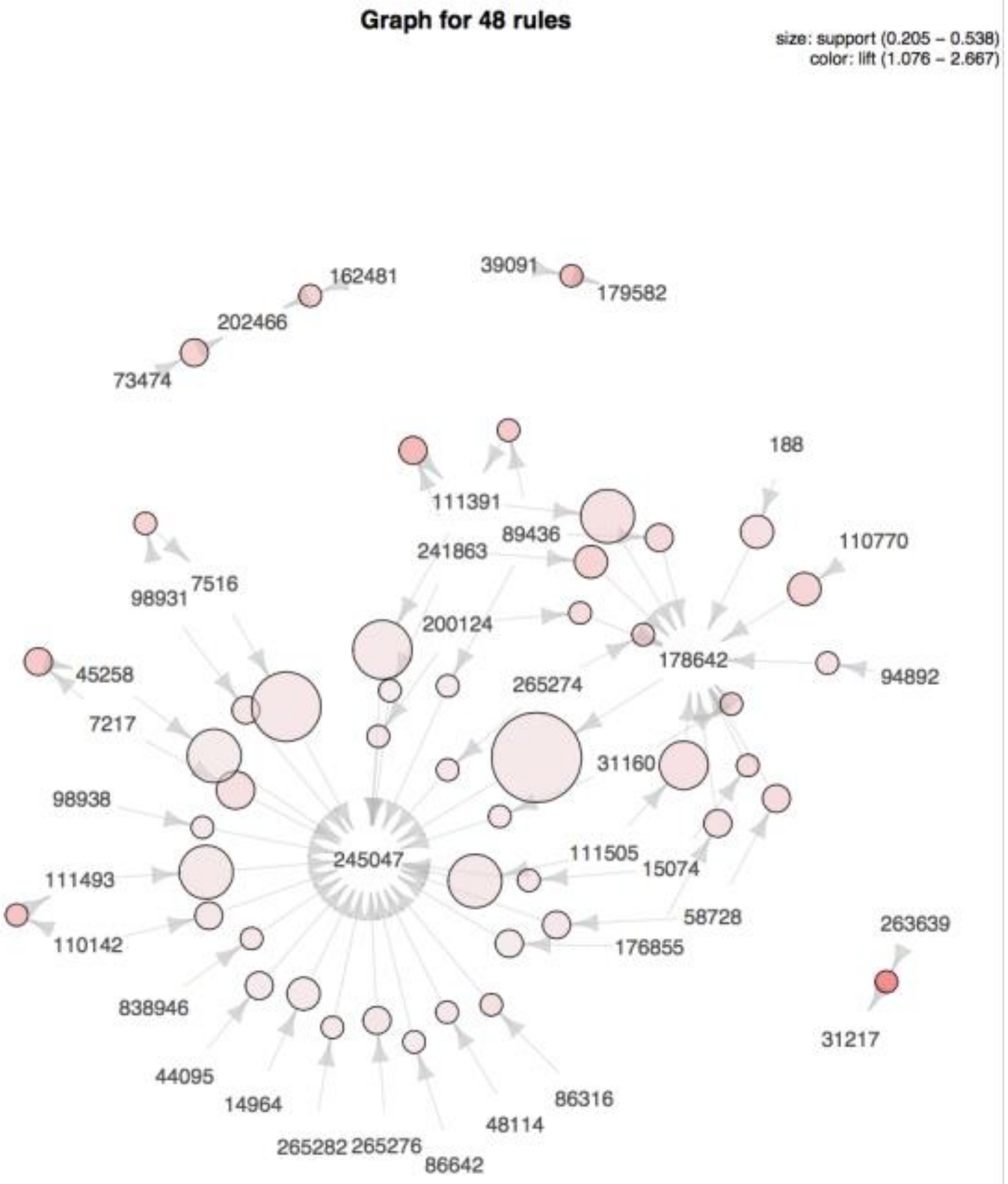


Figure 18 Rules plot of weight class #2 in scenario 1

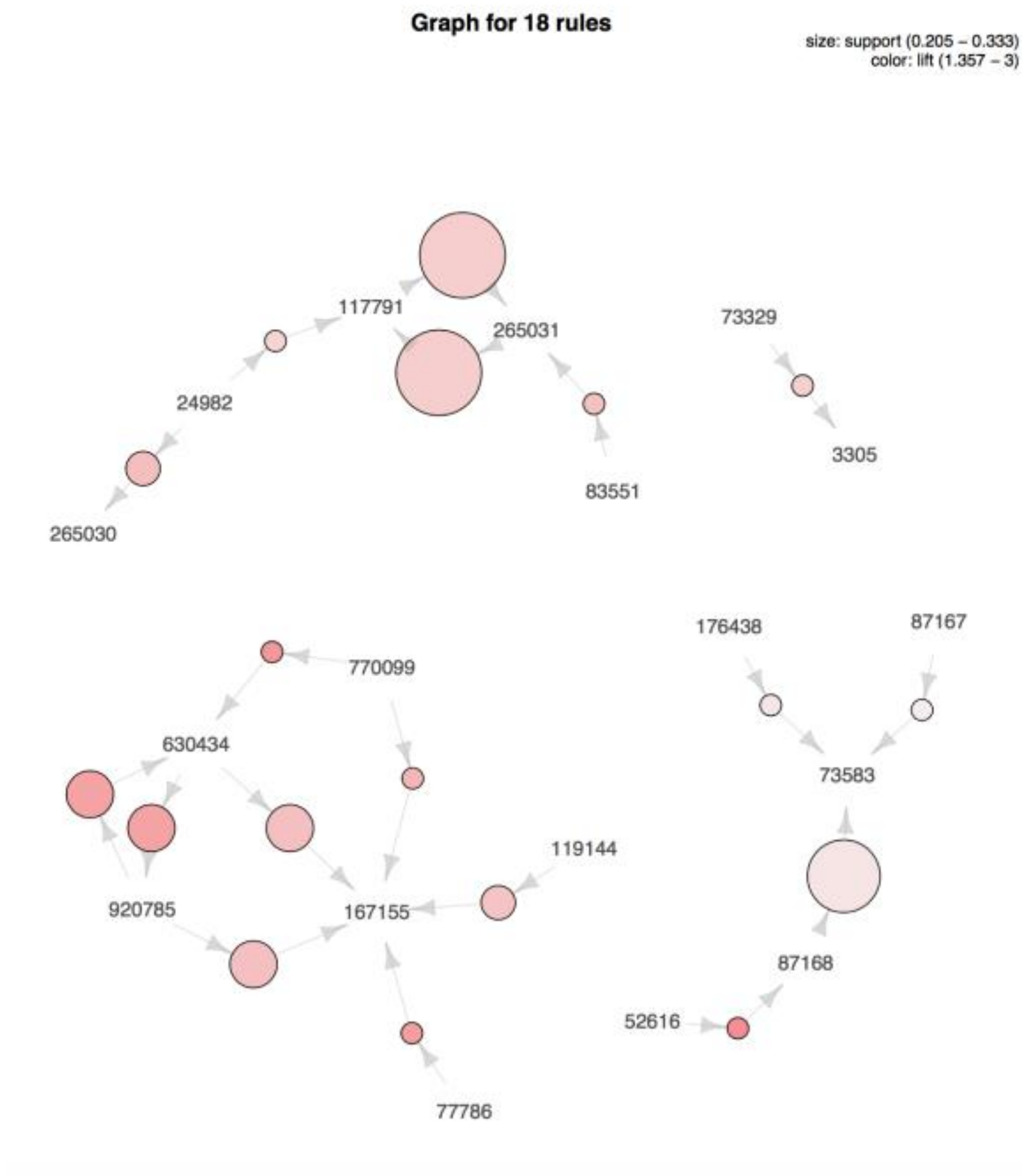


Figure 19 Rules plot of weight class #3 in scenario 1

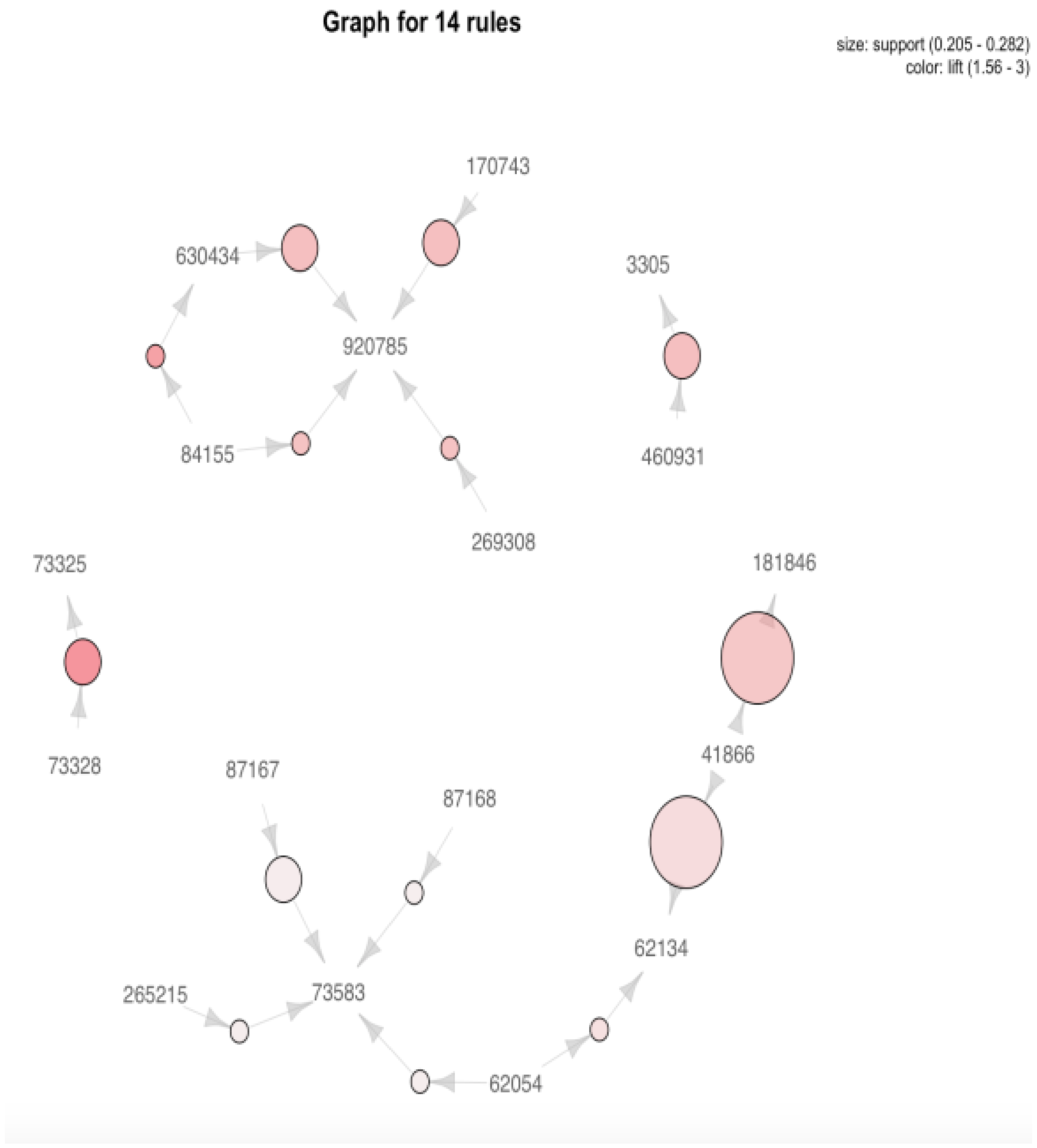


Figure 20 Rules plot of weight class #1 in scenario 2

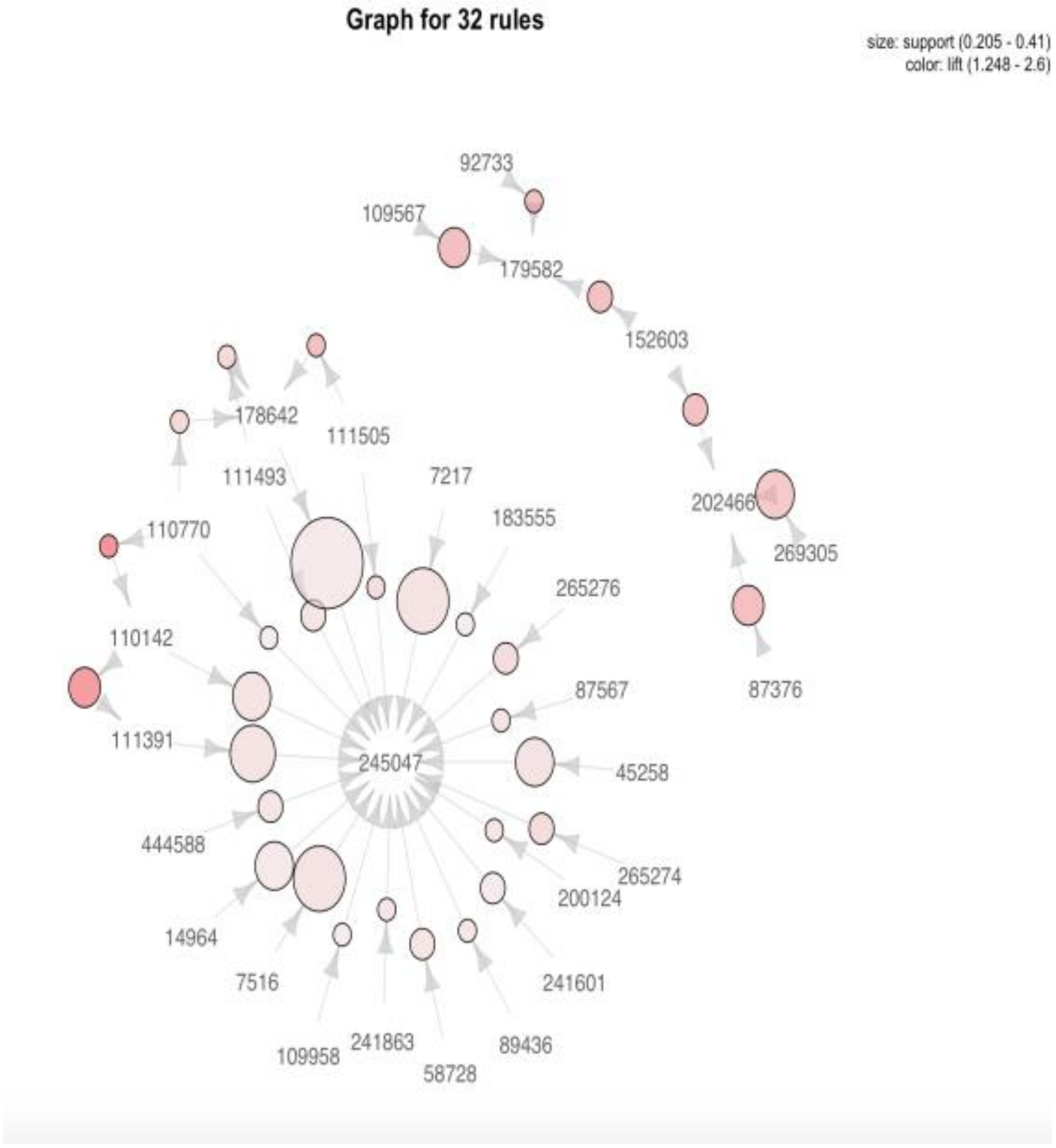


Figure 21 Rules plot of weight class #2 in scenario 2

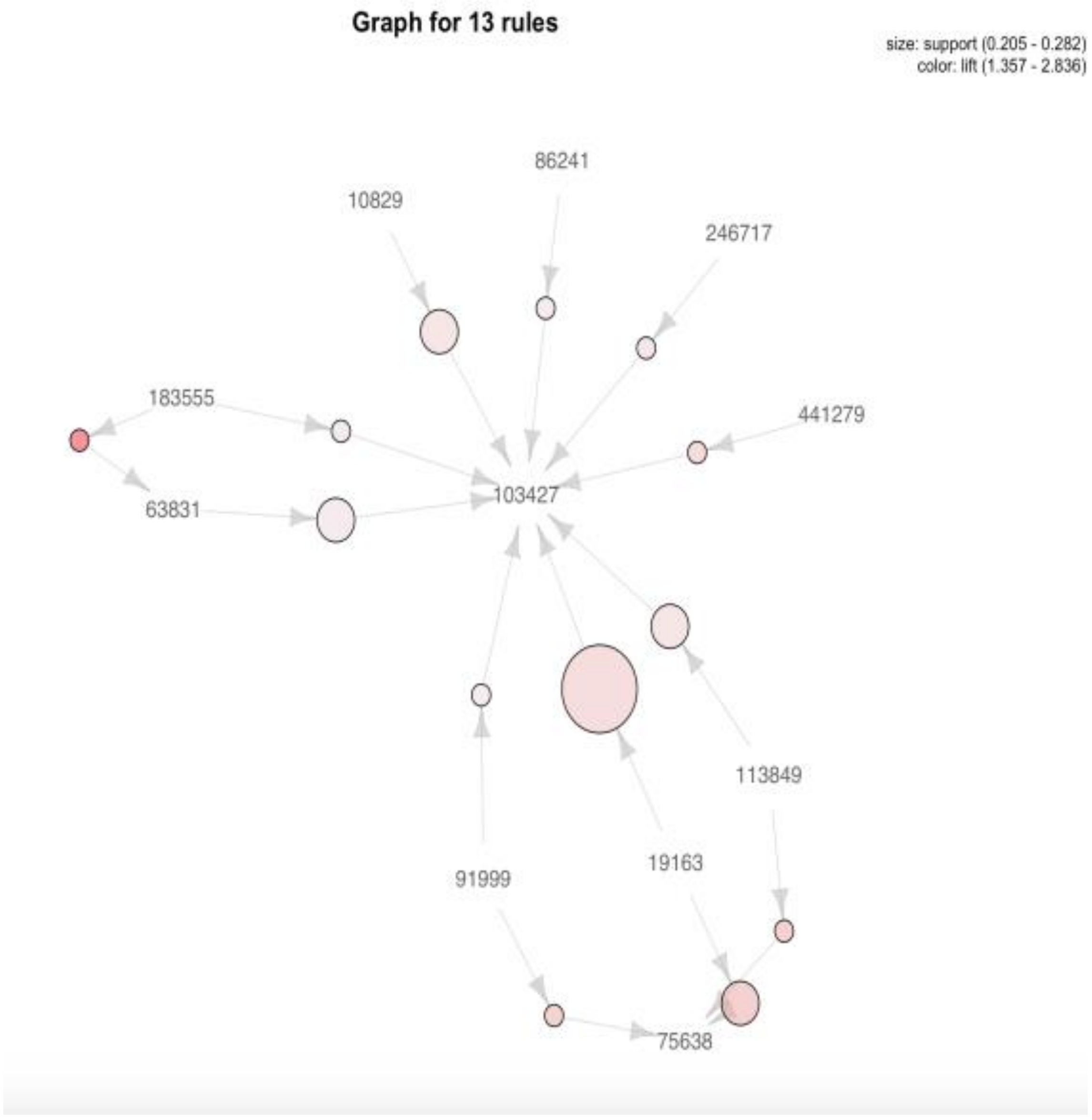


Figure 22 Rules plot of weight class #3 in scenario 2

The rules are created based on the ARM algorithm and each rule is considered as a pair of SKUs. SKUs included in each pair are reallocated to the location next to each other by implementing Exchange Method which will be introduced in next section.

4.4.5 Exchange Heuristic

In this section, Exchange Heuristic is applied within each weight-based class to reallocate the consequences to the location right after its antecedents in Stage #1 layout which is obtained from section 5.4.

In Scenario 1, 56 unique SKUs are impacted by implementing Exchange Method, the involved SKUs and their sequences in random, stage #1 and stage #2 layout are shown in the Tables 17 (a)(b)(c) below. The new locations of these SKUs depend on the location of the first SKU which has highest frequency in each weight-based class. The paired set in each weight-based class is inserted into the locations after the first SKU in the related class. The same process is done to scenario 2 with 56 unique SKUs as well. See Tables 18(a), (b) and (c).

The final output of this phase is the improved Stage #2 layout which is ready to use for computer simulation in the next section.

Table 17 (a) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 1

		SKU Number	**Frequency**	**Random Layout**	**Stage #1 Layout**	**Stage #2 Layout**
Class #1	1	117791	16	2228	3	3
	2	265031	15	2023	8	4
	3	87168	13	1630	16	5
	4	630434	12	1982	19	6
	5	920785	12	2189	20	7
	6	119144	11	1930	26	8
	7	24982	10	711	34	9
	8	73329	10	657	36	10
	9	87167	10	2209	37	11
	10	83551	9	1489	46	12
	11	176438	9	1664	52	13
	12	770099	9	1766	54	14
	13	52616	8	2442	62	15
	14	77786	8	1238	67	16

Table 17 (b) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 1

		SKU Number	Frequency	Random Layout	Stage #1 Layout	Stage #2 Layout
Class #2	15	178642	24	588	787	787
	16	7516	19	1678	788	788
	17	45258	17	1563	790	789
	18	111391	17	1381	791	790
	19	111493	16	239	793	791
	20	111505	16	1477	794	792
	21	188	12	628	798	793
	22	14964	12	1490	800	794
	23	7217	11	1663	803	795
	24	44095	11	1894	804	796
	25	73474	11	2485	805	797
	26	176855	11	2126	809	798
	27	39091	10	967	810	799
	28	58728	10	2481	811	800
	29	86642	10	1135	812	801
	30	89436	10	2151	813	802
	31	94892	10	1105	814	803
	32	98931	10	1339	815	804
	33	110142	10	718	816	805
	34	110770	10	1191	817	806
	35	162481	10	917	818	807
	36	241863	10	287	819	808
	37	265276	10	474	820	809
	38	15074	9	1192	822	810
	39	31160	9	1984	824	811
	40	48114	9	1797	825	812
	41	98938	9	1581	830	813
	42	200124	9	134	832	814
	43	263639	9	1682	834	815
	44	265274	9	124	835	816
	45	265282	9	2213	836	817
	46	838946	9	963	837	818
	47	86316	8	1721	848	819

Table 17 (c) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 1

		SKU Number	Frequency	Random Layout	Stage #1 Layout	Stage #2 Layout
Class #3	48	83626	13	2132	1633	1633
	49	241880	11	1405	1643	1634
	50	249707	11	1949	1644	1635
	51	10829	10	2537	1646	1636
	52	143789	10	1087	1652	1637
	53	271586	10	1740	1654	1638
	54	113849	9	986	1659	1639
	55	117091	9	95	1660	1640
	56	189145	9	1498	1662	1641

Table 18 (a) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 2

		SKU Number	Frequency	Random Layout	Stage #1 Layout	Stage #2 Layout
Class #1	1	73583	20	925	1	1
	2	62134	19	1236	2	2
	3	181846	14	354	3	3
	4	41866	13	938	6	4
	5	73325	13	229	7	5
	6	920785	13	2442	12	6
	7	3305	13	2371	15	7
	8	630434	11	500	23	8
	9	460931	11	1332	22	9
	10	170743	11	2111	20	10
	11	87167	11	550	16	11
	12	269308	10	2013	37	12
	13	62054	10	2300	32	13
	14	265215	10	1091	36	14
	15	87168	10	2155	34	15

Table 18 (b) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 2

		SKU Number	Frequency	Random Layout	Stage #1 Layout	Stage #2 Layout
Class #2	16	245047	25	910	806	806
	17	178642	19	2276	807	807
	18	202466	17	1712	809	808
	19	179582	17	2088	808	809
	20	7217	14	1106	810	810
	21	7516	14	996	811	811
	22	269305	13	363	818	812
	23	111391	13	457	817	813
	24	14964	13	1764	815	814
	25	110142	12	648	821	815
	26	45258	12	131	819	816
	27	87376	11	1159	822	817
	28	109567	11	823	823	818
	29	241601	11	1644	824	819
	30	152603	10	2261	832	820
	31	111493	10	1660	831	821
	32	444588	10	1018	834	822
	33	58728	10	1983	827	823
	34	109958	10	552	829	824
	35	84155	10	2275	833	825
	36	92733	9	2578	840	826
	37	265276	9	1411	847	827
	38	265274	9	349	846	828
	39	87567	9	2720	838	829
	40	200124	9	707	843	830
	41	89436	9	1457	839	831
	42	241863	9	2173	844	832
	43	73328	9	1533	845	833
	44	111505	8	1542	855	834

Table 18 (c) Locations of SKUs in random, stage #1, and stage #2 layout in scenario 2

		SKU Number	**Frequency**	**Random Layout**	**Stage #1 Layout**	**Stage #2 Layout**
Class #3	45	103427	23	2179	1708	1708
	46	75638	16	1155	1711	1709
	47	19163	11	1572	1719	1710
	48	63831	11	295	1720	1711
	49	110770	10	953	1830	1712
	50	183555	10	326	1833	1713
	51	10829	10	197	1723	1714
	52	86241	10	106	1728	1715
	53	91999	10	455	1729	1716
	54	113849	10	2643	1730	1717
	55	246717	9	2493	1743	1718
	56	441279	8	2183	1764	1719

In the tables above, all the SKUs involved in exchange heuristic which are the unique SKUs in association rules are sorted by frequency, the locations are shown in different phases, the original layout is random, stage #1 layout is the obtained in ranking and sorting phase, and stage #2 layout is the final layout by implementing exchange heuristic. It's obvious that the correlated SKUs are moving closer the I/O point as well as closer to each other which further lead to a reduction in picking up activity.

4.4.6 Computer Simulation

In this section, a static simulation model is developed in AutoMod to compare the performance in two scenarios, respectively, random layout and improved layout by implementing the proposed method in this thesis. The measurement of outperformance is average pickup time. Appendix 5 shows the code in AutoMod.

4.4.6.1 Output Analysis

The output from simulation indicates that the improvement is significant by implementing the method proposed in this thesis. By comparing random layout to Stage #1 and Stage #2 layouts, the grand average pickup time decrease from 308603.9.5 to 39740 then to 34848.7 seconds while the trip time decreases from 309591.9 to 41006.1 then to 36106 seconds. The improvement rate from random to Stage #1 is 87.12% and 12.31% from Stage #1 to Stage #2 in term of pickup time, the total improvement is 88.71%. In term of trip time, the improvements are 86.75%, 11.95% and 88.34%. See Table 19-20 and Figure 23-35. In other words, labor cost in the order picking process is reduced by almost 90%. This heuristic is a very good attempt of building a new DC from the scratch or rebuilding an existing DC because the solution is relatively quick and easy to implement.

Table 19 Numeric improvement overview

Scenario 1

	pickup time			**trip time**		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
Average	308682.5	38597.2	34535.2	309664.9	40144.8	36082.8
Std. Dev	43316.5	2469.4	2052.7	43258.0	2653.1	2160.9
Max	370859.7	43064.6	36928.0	371656.2	44665.6	38565.1
Min	229049.9	31603.5	29382.2	230827.3	33060.9	31242.6

Scenario 2

	pickup time			**trip time**		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
Average	308525.3	40900.8	35162.1	309518.9	41867.4	36129.2
Std. Dev	45721.6	3203.8	3002.8	45717.9	3089.1	2819.8
Max	370739.8	44528.2	37878.3	371536.2	45205.0	38565.1
Min	231640.1	30256.2	25894.8	232516.5	31479.3	27118.0

Table 20 Percentage improvement overview

Pickup time

		Scenario1			Scenario2		
		Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
Scenario 1	**Random**	0.00%	-	-	-	-	-
	Stage #1	87.50%	0.00%	-	-	-	-
	Stage #2	88.81%	10.52%	0.00%	-	-	-
Scenario 2	**Random**	-	-	-	0.00%	-	-
	Stage #1	-	-	-	86.74%	0.00%	-
	Stage #2	-	-	-	88.60%	14.03%	0.00%

Trip time

		Scenario1			Scenario2		
		Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
Scenario 1	**Random**	0.00%	-	-	-	-	-
	Stage #1	87.04%	0.00%	-	-	-	-
	Stage #2	88.35%	10.12%	0.00%	-	-	-
Scenario 2	**Random**	-	-	-	0.00%	-	-
	Stage #1	-	-	-	86.47%	0.00%	-
	Stage #2	-	-	-	88.33%	13.71%	0.00%

Comparison of pickup and trip time in each order picking operation is shown in Table 21-22 and Figures 23-35 below.

Table 21 Pickup time and trip time in each order in random layout and stage #1 & #2 layout of scenario 1

Scenario 1

	pickup time			**trip time**		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
order1	274,985.5	39,983.3	36,650.7	275,882.9	41,717.9	38,385.1
order2	235,021.3	31,603.5	29,785.2	235,817.8	33,060.9	31,242.6
order3	313,026.4	39,275.8	36,002.4	314,590.5	40,285.6	37,012.2
order4	240,569.1	39,290.5	36,011.1	241,338.9	40,153.6	36,874.2
order5	365,635.9	39,922.6	35,186.8	365,779.0	41,860.6	37,124.2
order6	350,083.0	37,346.3	31,209.5	351,266.2	38,977.0	32,840.2
order7	345,435.7	37,708.9	34,429.5	346,045.5	38,958.8	35,679.3
order8	287,162.5	38,701.9	35,369.1	288,473.3	40,343.0	37,010.2
order9	306,332.9	41,287.0	33,748.6	307,116.0	43,194.7	35,656.3
order10	365,575.1	36,791.7	34,914.2	366,685.9	37,574.9	35,697.3
order11	352,848.0	34,116.9	29,382.2	354,758.8	36,161.0	31,426.3
order12	245,521.9	37,094.3	33,820.9	246,158.4	38,788.8	35,515.3
order13	357,851.9	40,148.4	35,473.1	358,381.8	41,829.5	37,154.2
order14	313,406.2	41,559.3	36,824.7	314,269.4	43,173.7	38,439.1
order15	277,094.1	41,462.3	36,727.7	277,964.8	43,116.7	38,382.1
order16	365,589.0	41,285.3	36,550.7	367,393.1	43,299.7	38,565.1
order17	327,011.4	38,635.9	33,901.2	327,714.6	40,397.0	35,662.3
order18	333,415.4	36,763.1	32,028.4	333,705.2	37,586.2	32,851.5
order19	316,213.7	36,608.7	34,790.5	316,876.8	37,458.5	35,640.3
order20	325,035.6	40,091.1	36,758.3	325,965.4	41,798.9	38,466.1
order21	282,645.6	39,868.1	36,535.3	284,409.7	41,735.9	38,403.1
order22	245,665.7	37,694.6	33,019.3	246,789.7	38,757.8	34,082.4
order23	275,087.9	33,621.3	31,743.7	276,291.9	34,431.1	32,553.5
order24	337,699.8	37,080.3	32,345.7	338,469.6	38,991.1	34,256.4
order25	229,049.9	34,714.2	32,895.9	230,827.3	35,870.6	34,052.4
order26	279,306.4	36,870.0	34,992.1	279,462.8	38,793.8	36,916.2
order27	350,364.5	43,064.6	36,928.0	350,507.7	44,665.6	38,529.1
order28	244,047.8	41,232.0	36,497.3	245,491.9	43,059.7	38,325.1
order29	335,800.0	41,587.0	36,852.3	336,129.8	43,254.7	38,520.1
order30	296,296.0	36,377.1	31,642.4	297,353.4	37,520.2	32,785.5
order31	344,531.4	38,664.6	35,331.8	345,962.1	40,439.0	37,106.2
order32	314,689.8	39,988.8	35,254.1	315,600.6	41,789.9	37,055.2
order33	297,647.4	37,506.3	34,226.8	299,198.2	38,862.8	35,583.3
order34	370,859.7	41,657.0	36,922.3	371,656.2	43,284.7	38,550.1
order35	355,992.8	39,789.8	35,055.1	357,676.9	41,870.9	37,136.2

	pickup time			trip time		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
order36	302,029.3	38,531.6	32,395.0	302,892.5	40,309.0	34,172.4
order37	234,012.0	39,897.8	33,761.2	235,002.7	41,618.9	35,482.3
order38	290,917.9	37,536.9	34,257.5	292,081.9	38,826.8	35,547.3
order39	354,159.5	39,932.1	36,652.7	354,942.7	41,826.5	38,547.1

Table 22 Pickup time and trip time in each order in random layout and stage #1 & #2 layout of scenario 2

Scenario2

	pickup time			**trip time**		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
order1	231,640.1	39,106.5	34,798.5	232,516.5	39,793.3	35,485.3
order2	285,661.0	42,548.6	36,299.4	287,078.4	43,235.4	36,986.2
order3	278,703.8	42,846.3	36,170.4	279,707.8	43,616.1	36,940.2
order4	242,477.1	41,817.3	37,082.6	244,121.2	43,050.7	38,316.1
order5	367,170.2	42,129.9	35,880.7	368,193.3	43,366.4	37,117.2
order6	362,659.1	40,858.1	32,780.3	363,625.9	42,419.2	34,341.4
order7	242,145.7	43,402.6	36,726.7	242,328.9	44,977.0	38,301.1
order8	244,041.4	33,703.9	27,940.7	244,677.9	35,574.6	29,811.4
order9	354,714.9	43,034.9	36,359.1	355,378.0	43,815.1	37,139.2
order10	241,269.4	38,521.5	34,160.2	242,993.5	39,861.6	35,500.3
order11	370,739.8	38,460.6	32,201.4	371,536.2	39,200.8	32,951.5
order12	338,484.2	42,886.6	36,637.3	339,267.4	44,754.3	38,505.1
order13	359,703.4	38,534.3	31,858.4	360,459.9	39,557.4	32,881.5
order14	298,024.0	42,593.6	36,344.4	299,508.0	43,280.4	37,031.2
order15	348,945.4	44,528.2	37,842.3	349,181.9	45,205.0	38,529.1
order16	329,026.4	40,741.7	36,380.4	330,110.5	41,428.5	37,067.2
order17	285,183.7	43,974.5	37,725.3	286,814.5	44,661.3	38,412.1
order18	360,572.6	42,694.3	36,445.1	362,150.0	43,394.4	37,145.2
order19	329,342.7	39,131.2	32,455.3	329,699.1	40,972.3	34,296.4
order20	267,109.5	36,133.1	29,937.2	267,386.0	37,329.5	31,133.6
order21	283,535.0	41,112.7	36,308.4	285,139.0	41,799.5	36,995.2
order22	285,680.0	43,838.2	37,588.9	286,583.2	44,658.3	38,409.1
order23	273,134.1	30,256.2	25,894.8	274,130.6	31,479.3	27,118.0
order24	236,917.1	34,825.5	31,919.4	237,833.5	35,515.3	32,609.2
order25	266,102.2	39,382.9	31,305.1	267,092.9	40,800.3	32,722.5
order26	340,794.7	37,766.3	30,115.2	342,678.8	39,119.8	31,468.6
order27	368,179.5	44,127.5	37,878.3	368,536.0	44,814.3	38,565.1
order28	349,752.7	42,212.6	37,851.3	351,156.8	42,899.4	38,538.1
order29	313,683.1	42,506.6	36,257.4	315,087.1	43,313.4	37,064.2
order30	354,119.9	42,692.6	36,443.4	356,094.3	43,379.4	37,130.2
order31	273,594.7	37,765.3	36,320.4	274,017.8	38,452.1	37,007.2
order32	326,641.2	41,213.4	36,419.4	326,957.7	41,900.2	37,106.2
order33	260,900.4	39,467.2	37,579.3	262,197.8	40,234.0	38,346.1
order34	332,215.9	43,561.8	37,312.6	333,206.7	44,715.3	38,466.1
order35	367,421.4	43,155.3	36,479.4	368,564.6	43,842.1	37,166.2

	pickup time			trip time		
	Random	**Stage #1**	**Stage #2**	**Random**	**Stage #1**	**Stage #2**
order36	342,646.2	44,079.5	37,830.3	343,442.7	44,766.3	38,517.1
order37	277,585.0	42,905.6	36,229.7	278,328.2	43,659.1	36,983.2
order38	281,004.2	43,950.5	37,701.3	281,627.4	44,637.3	38,388.1
order39	360,964.3	42,664.6	37,860.3	361,827.4	43,351.4	38,547.1

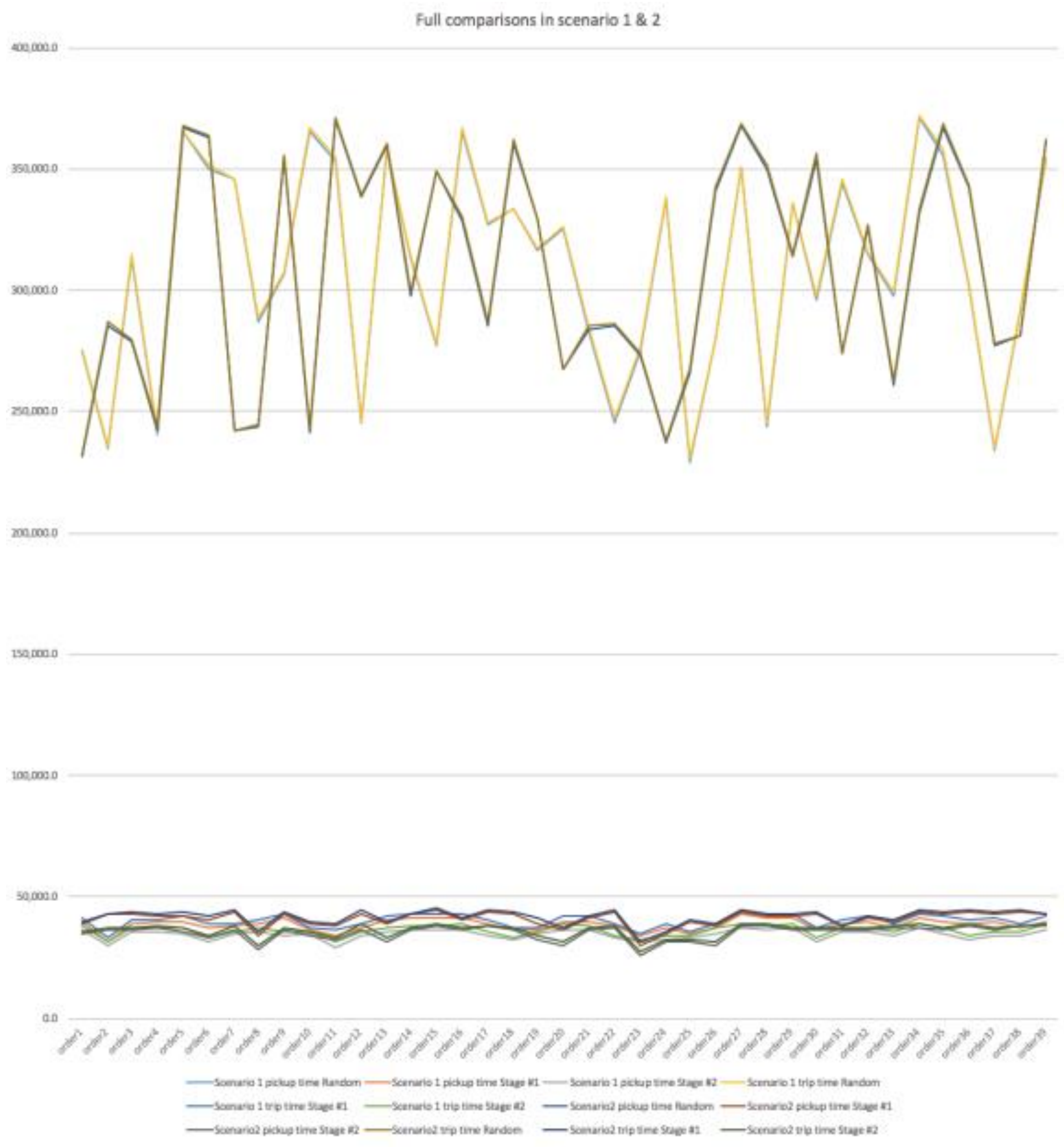


Figure 23 Full comparisons of scenario 1 & 2

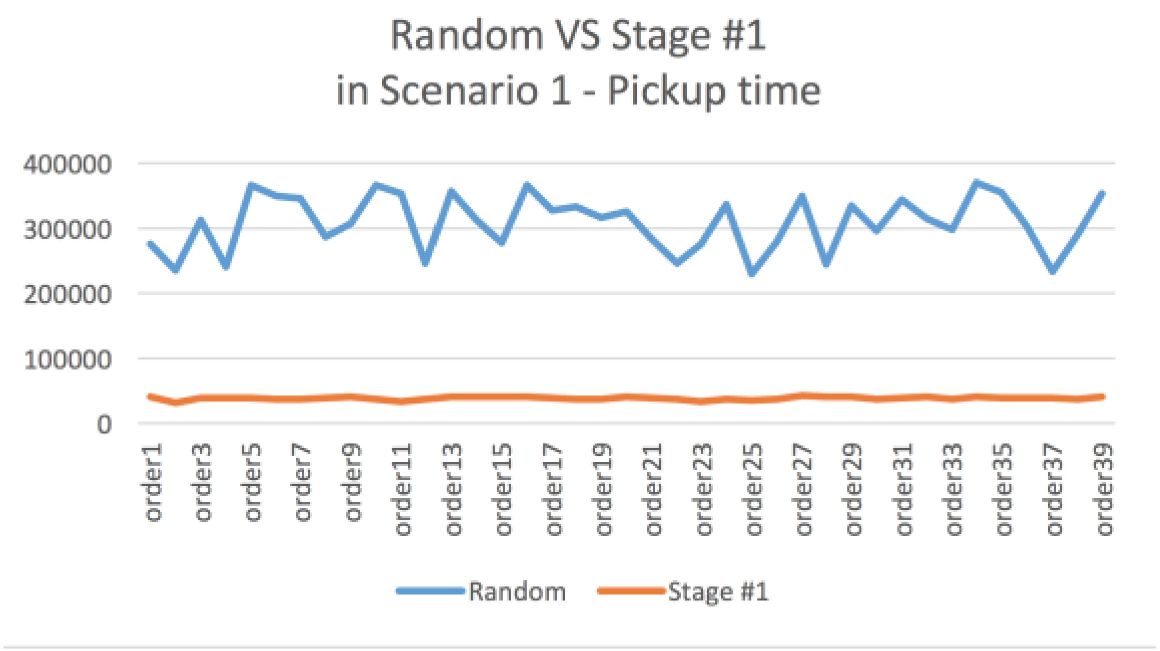


Figure 24 Pickup time of random VS stage #1 in scenario 1

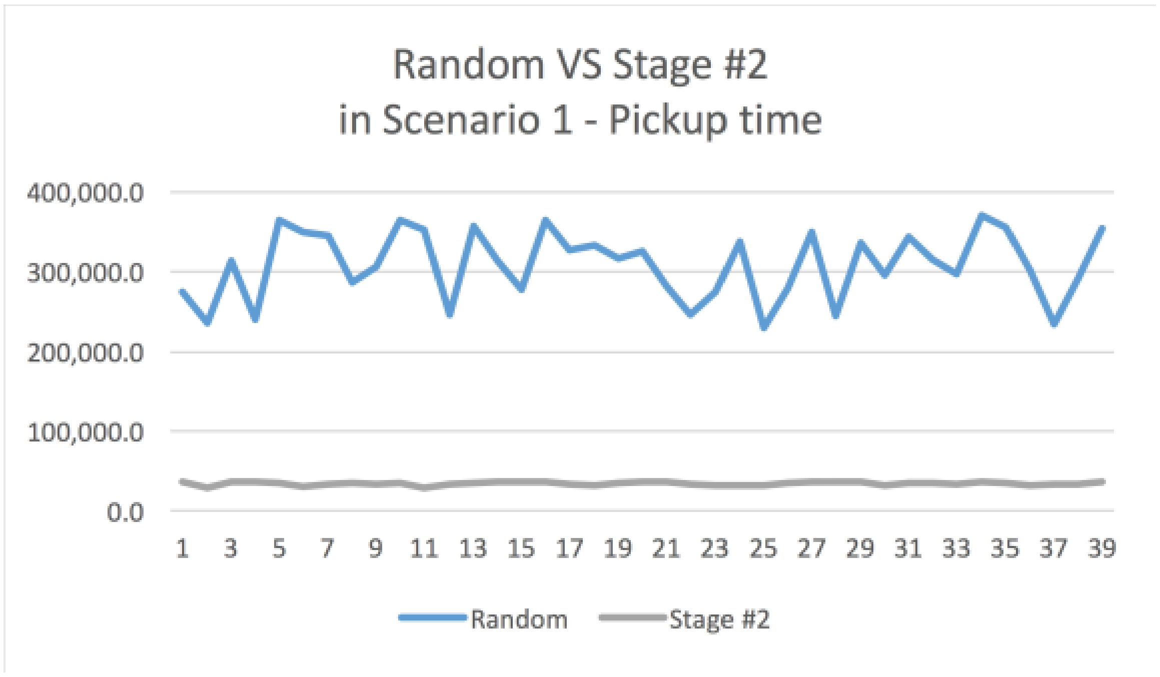


Figure 25 Pickup time of random VS stage #2 in scenario 1

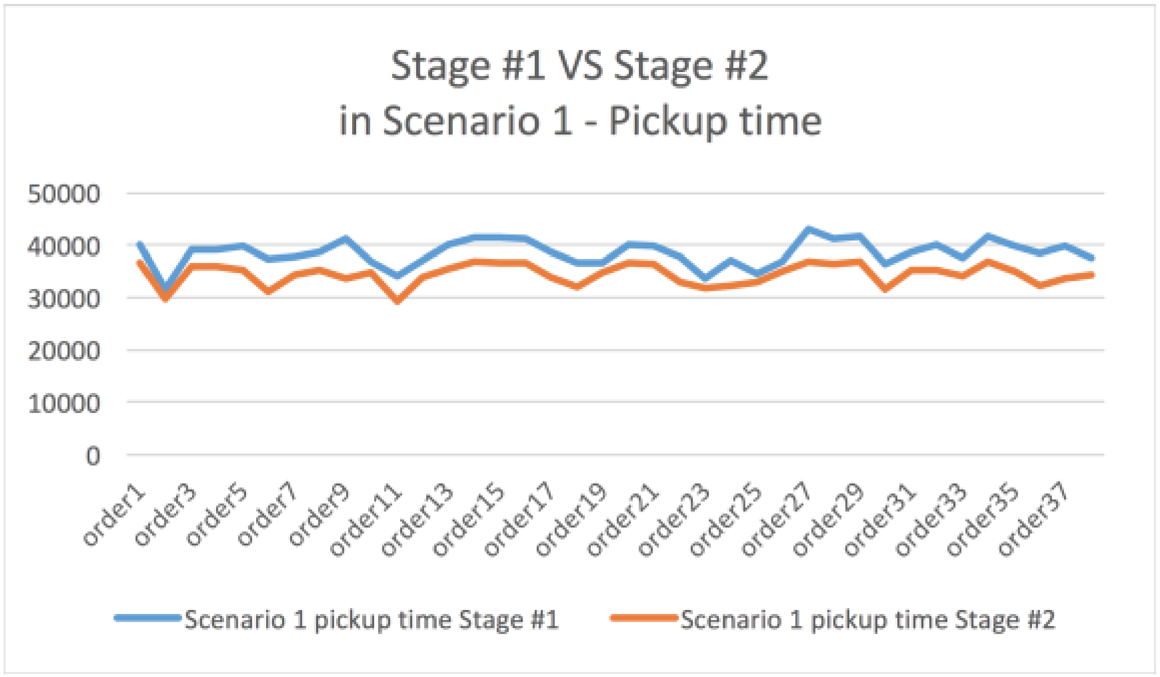


Figure 26 Pickup time of stage #1 VS stage #2 in scenario 1

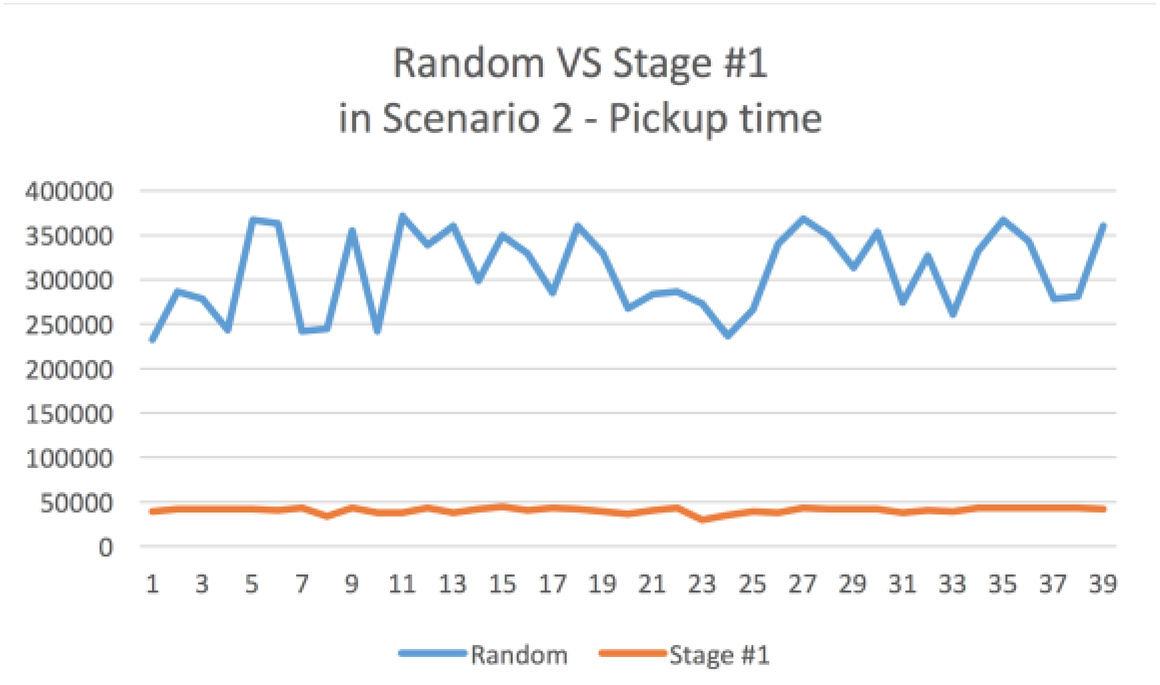


Figure 27 Pickup time of random VS stage #1 in scenario 2

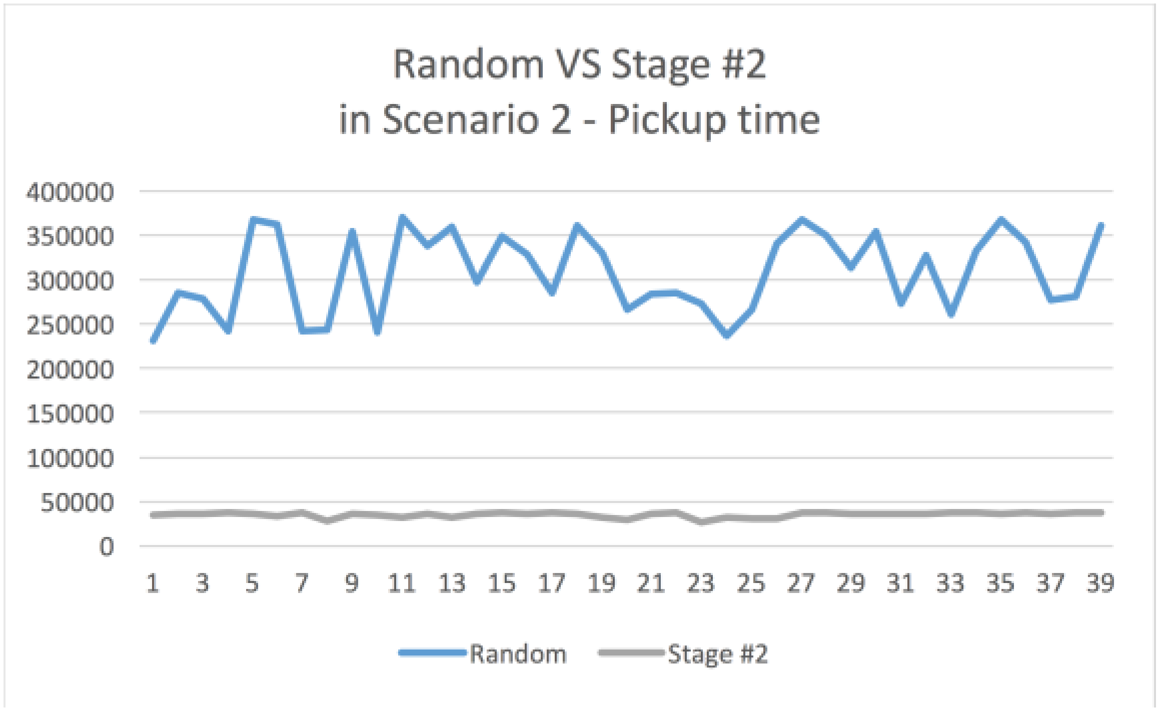


Figure 28 Pickup time of random VS Stage #2 in scenario 2

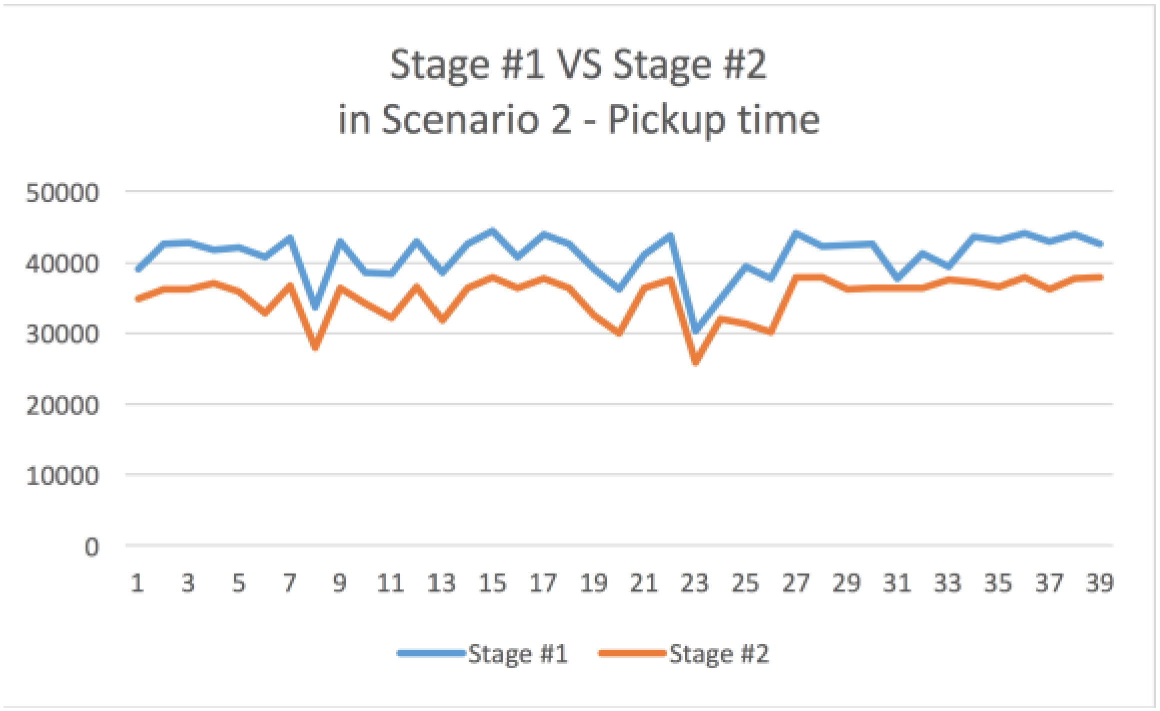


Figure 29 Pickup time of stage #1 VS stage #2 in scenario 2

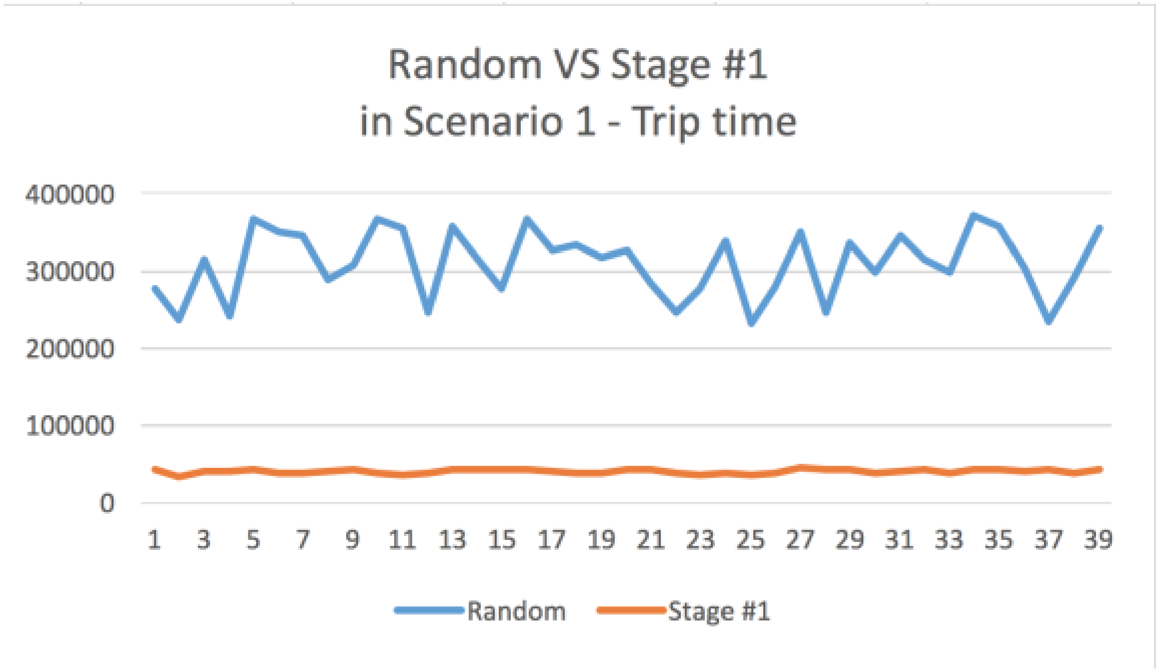


Figure 30 Trip time of random VS stage #1 in scenario 1

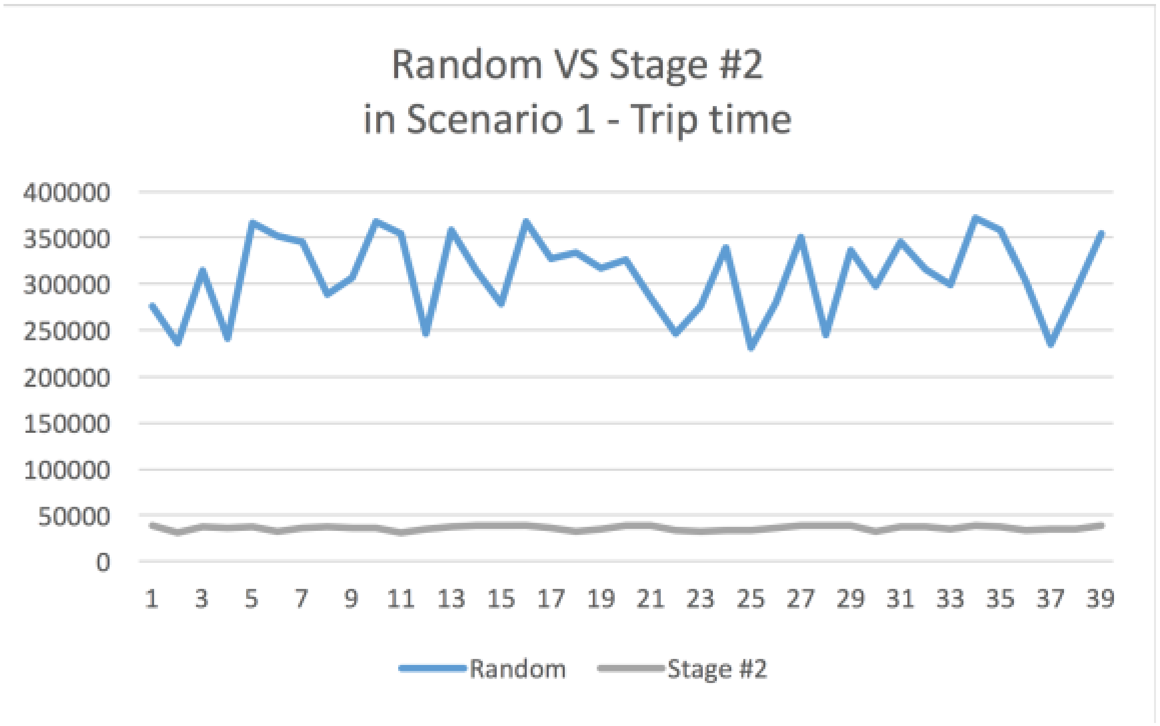


Figure 31 Trip time of random VS stage #2 in scenario 1

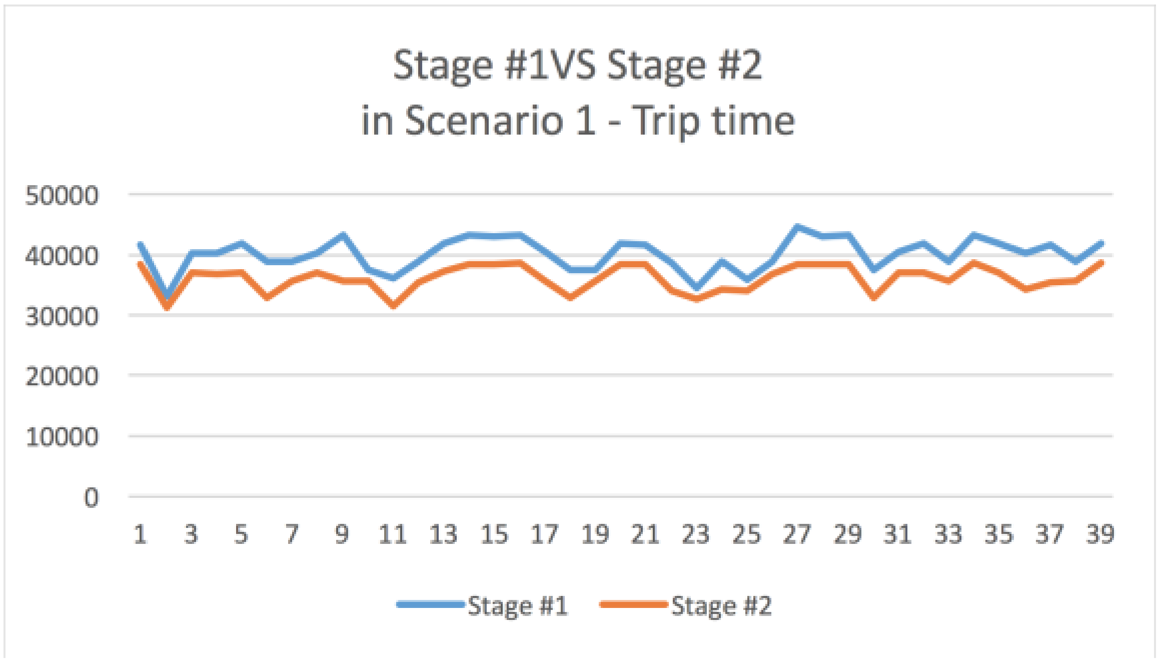


Figure 32 Trip time of stage #1 VS stage #2 in scenario 1

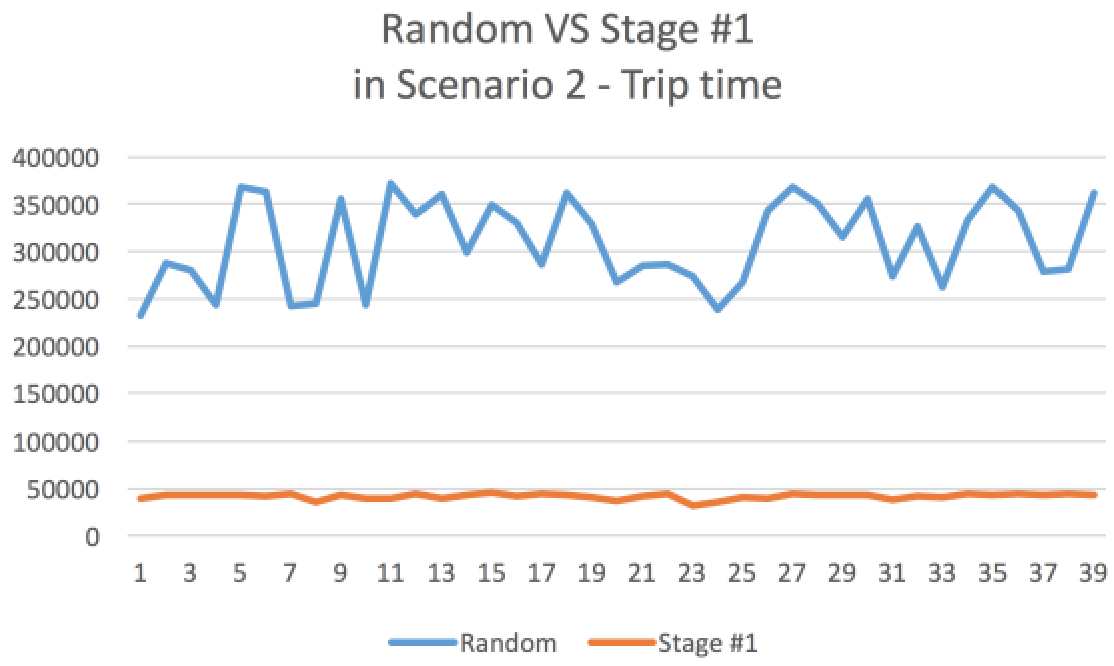


Figure 33 Trip time of random VS stage #1 in scenario 2

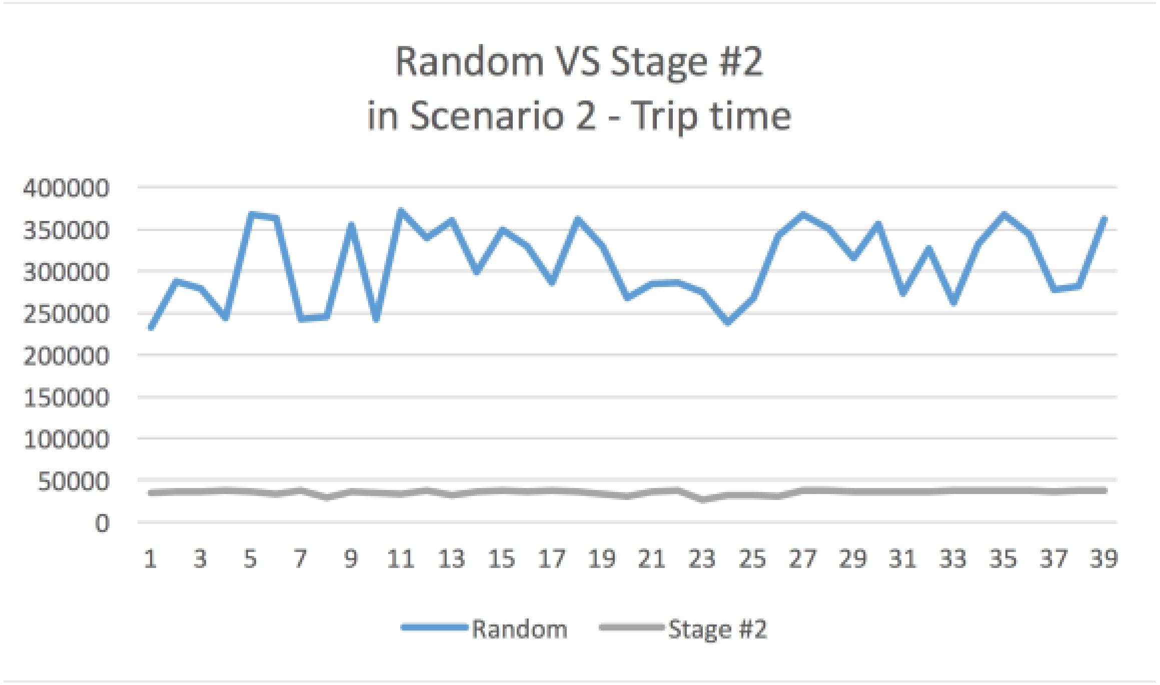


Figure 34 Trip time of random VS stage #2 in scenario 2

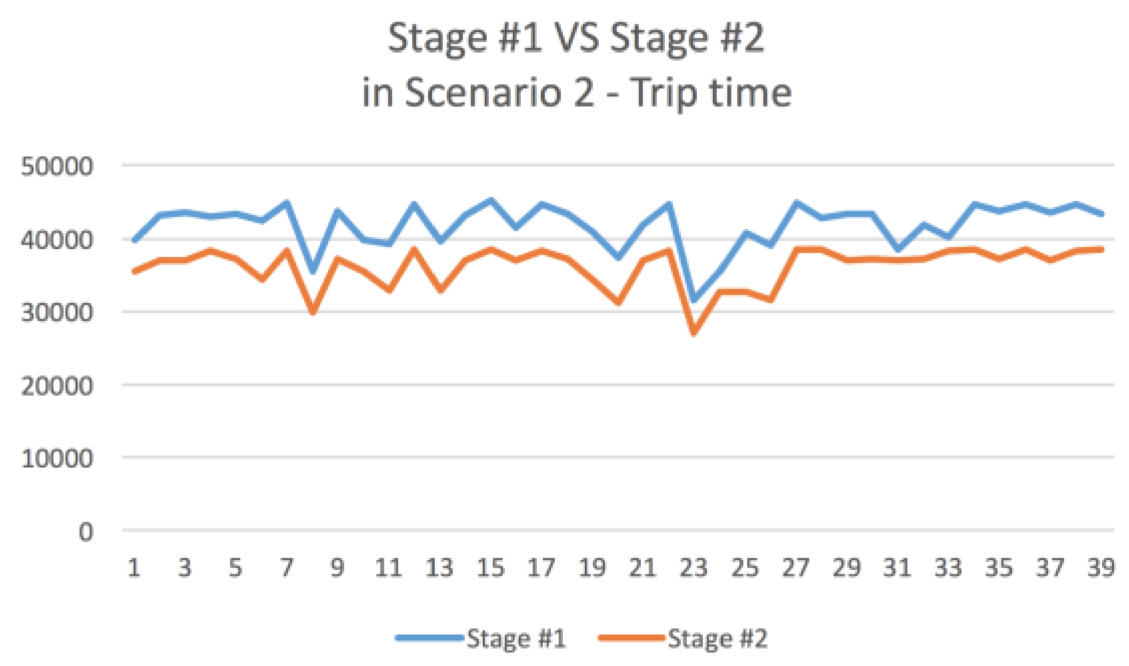


Figure 35 Trip time of stage #1 VS stage #2 in scenario 2

V. CONCLUSION AND FUTURE RESEARCH

As described in Section 1.3 for objectives of this thesis, the case study presents a quick solution of storage location assignment problem in a practical level with significant improvement by implementing the proposed heuristic as well as the combination of picker-to-parts pickup method and S-shape routing method.

5.1 Conclusion

The results of this thesis indicate that the combination of picker-to-parts method and S-shape routing selection method has positive impact on improving efficiency of order picking activity. The reason for choosing S-shape routing method is that it is commonly used in practice, and the selection of S-shape routing method reduced the complexity of the problem, and thus, shorten the computational time.

By balancing the computational time and output between optimal and heuristic solutions, optimal solution is not implementable because of the huge computational time. As is mentioned in section 4.3, the computational time for obtaining optimal solution is too long to meet our goal of producing a quick solution, especially when number of SKUs increases. In term of performance, the proposed heuristic is 6.7% worse than the optimal in the experiment involved 10 SKUs. We can conclude that the proposed heuristic is implementable in practice because it provides a good solution in much shorter time.

The proposed heuristic integrates techniques such as data analysis, association mining, mathematical formulation and computer simulation. Five steps of the proposed heuristic ensure the pickup priority accounting for weight and frequency of SKUs. Also, the heuristic improves the layout design by exchanging the storage locations of SKUs based on the associations between SKUs generated via association rules mining. The numerical results from different scenarios show that the average improvement of 87.12% from random layout to Stage #1 layout and 12.31% from Stage #1 layout to Stage #2 layout in term of pickup time. In term of pickup time, the improvements are 86.75% from random layout to stage #1 layout and 11.95% from stage #1 layout to stage #2 layout.

ARM algorithm is an important technique to identify correlations between SKUs among different transactions. And it has more potential in improving order picking operation by considering more rules. Before implementing the proposed heuristic in practice, an appropriate combination of support and confidence in ARM algorithm needs to be determined in advanced, since it is the key activity to balance the outperformance and computation time. According to the result form design of experiment, the selected combination of support (0.2) and confidence (0.8) is significant to impact number of rules produced via ARM.

Furthermore, simulation provides an easy and convenient way to visualize and analysis the ordering picking operations. C language-based coding environment in AutoMod makes the model easy to be expanded and modified to test more experiments. In addition, simulation is also an efficient way to validate and verify the method applied and control the potential risk by analyzing the simulation output.

The heuristic proposed in this thesis is a ready-to-use “package” for layout design, the only input for such package is the raw data set with transaction numbers, SKU numbers and weights of SKUs. It takes much shorter time to produce a good solution when compare to optimal solution. It is recommended to implement such heuristic for new layout design and/or layout improvement in an existing DC.

5.2 Future Research

Due to limited time, this thesis only provides comparison among two layouts generated from the proposed heuristics and randomized layout. The future research may focus on: First, applying proposed heuristic into new layout, for example, fishbone layout to achieve better performance. Second, identifying the break-even point of support and confidence regarding specific data set in ARM phase to produce the best solution of number of rules. Finally, comparing the proposed method to other existing algorithm to indicate the performance, like Genetic Algorithm and Ant Colony Optimization algorithm.

APPENDIX SECTION

APPENDIX 1: AMPL CODE OF QAP FORMULATION

```
param n:=20;
param f{1..n,1..n};
param d{1..n,1..n};
param c{1..n,1..n};
var x{1..n,1..n} integer;
 minimize obj_function: sum{i in 1..n} sum{j in 1..n} sum{k in 1..n} sum{l in 1..n}
f[i,j]*d[k,l]*x[i,k]*x[j,l] + sum{i in 1..n} sum{j in 1..n} c[i,j]*x[i,j];
 subject to constr1{j in 1..n}: sum{i in 1..n} x[i,j] = 1;
 subject to constr2{i in 1..n}: sum{j in 1..n} x[i,j] = 1;
 subject to bound1{i in 1..n,j in 1..n}: 0<=x[i,j]<=1;
 #subject to con_add{i in 1..n}: x[i,i]=0;
 data HCP13.5.1.dat; option solver cplexamp; solve; display x;
 display _total_solve_time
```

APPENDIX 2: R CODE OF DATA SAMPLE SELECTION

```
getwd()

setwd("/Users/yueli/Desktop/Thesis R code")

library("readxl")

library("dplyr")

library("stringr")

library("stringi")

dt_table <- read_excel("GM.xlsx","GM")

dt_ASGN <- dt_table[,1]

dt_ASGN <- unique(dt_ASGN)

## choose ASGN count between 150-250

df<-data.frame(dt_table$ASGN,dt_table$Product)

names(df)[1:2] <- c("Assignment", "Product")

group_df <- group_by(df,as.numeric(df$Assignment))

count_df<-count_(group_df)

final_df<-filter(count_df, count_df$n>= 150 & count_df$n <= 250)

names(final_df)[1:2] <- c("ASGN", "Frequency")

all_df<-filter(dt_table,dt_table$ASGN %in% final_df$ASGN)

write.csv(all_df, "150to250.csv")

## randomly choose 1/3 of orders with items between 150 and 250

dt_ASGN1 <- all_df[,1]

dt_ASGN1 <- unique(dt_ASGN1)

dt_ASGN_rand <- mutate(dt_ASGN1,rand_num = runif(nrow(dt_ASGN1),1,100))
```

```
dt_table_sort <- arrange(dt_ASGN_rand,as.numeric(desc(dt_ASGN_rand$rand_num)))

table1 <- dt_table_sort[1:ceiling(1/3 * nrow(dt_table_sort)),]

final_table_1 <- filter(all_df, all_df$ASGN %in% table1$ASGN)

write.csv(final_table_1, "onethirdof150to250.csv")
```

APPENDIX 3: R CODE OF WEIGHT-BASED CLASS AND STAGE #1 LAYOUT

```
quan <- quantile(final_table_1$Weight,probs = seq(0,1,0.33333333))
weight_filter_1 <- filter(final_table_1,final_table_1$Weight<=quan[2])
group_weight_1<-group_by(weight_filter_1,as.numeric(weight_filter_1$Product))
frequencyofclass1 <- count(group_weight_1)
dec_freq_class1 <- arrange(frequencyofclass1,as.numeric(desc(frequencyofclass1$n)))
names(dec_freq_class1)[1:2] <- c("Product", "Freq")
weight_filter_2 <- filter(final_table_1,final_table_1$Weight>=quan[2] &
final_table_1$Weight <= quan[3])
group_weight_2<-group_by(weight_filter_2,as.numeric(weight_filter_2$Product))
frequencyofclass2 <- count(group_weight_2)
dec_freq_class2 <- arrange(frequencyofclass2,as.numeric(desc(frequencyofclass2$n)))
names(dec_freq_class2)[1:2] <- c("Product", "Freq")
weight_filter_3 <- filter(final_table_1,final_table_1$Weight >= quan[3])
group_weight_3<-group_by(weight_filter_3,as.numeric(weight_filter_3$Product))
frequencyofclass3 <- count(group_weight_3)
dec_freq_class3 <- arrange(frequencyofclass3,as.numeric(desc(frequencyofclass3$n)))
names(dec_freq_class3)[1:2] <- c("Product", "Freq")
itemlocationall <- rbind(dec_freq_class3,dec_freq_class2,dec_freq_class1)
location<-c(1:nrow(itemlocationall))
itemlocationall <- cbind(itemlocationall,location)
write.csv(weight_filter_1, "weightclass1.csv")
write.csv(weight_filter_2, "weightclass2.csv")
write.csv(weight_filter_3, "weightclass3.csv")

write.csv(dec_freq_class1, "itemlocationclass1.csv")

write.csv(dec_freq_class2, "itemlocationclass2.csv")

write.csv(dec_freq_class3, "itemlocationclass3.csv")

write.csv(itemlocationall, "itemlocationall.csv")

## Stage 1 layout matrix

new_matrix <- matrix(ncol = nrow(itemlocationall), nrow = nrow(itemlocationall), c(0))

rownames(new_matrix) <- 1:nrow(itemlocationall) #item

colnames(new_matrix) <- 1:nrow(itemlocationall) #location

new_matrix

for (i in 1:nrow(new_matrix))

{

  new_matrix[itemlocationall$location[i],itemlocationall$location[i]] <-

itemlocationall$Product[i]

}

dim(new_matrix)


write.csv(new_matrix, "stage1layoutmatrix.csv")
```

APPENDIX 4: R CODE OF ASSOCIATION RULES MINING AND OUTPUT REFORMATION

```
## Association rules mining for class #1
install.package("arules")
library("arules")
HEBdata <- weight_filter_1
HEBdata<-transform(HEBdata, ASGN=as.factor(ASGN))
HEBdata<-transform(HEBdata, Product=as.factor(Product))
HEBdata<-transform(HEBdata, Aisle=as.factor(Aisle))
HEBdata<-transform(HEBdata, Location=as.factor(Location))
HEBdata<-transform(HEBdata, Description=as.factor(ItemDescription))
HEBdata<-transform(HEBdata, Weight=as.numeric(Weight))
length(levels(HEBdata$ASGN))
length(levels(HEBdata$Product))
quantile(HEBdata$Weight)
WC <- split(x=HEBdata[,"Description"], f=HEBdata$ASGN)
itemM <- split(x=HEBdata[,"Product"], f=HEBdata$ASGN)
WC <- lapply(WC, unique)
itemM <- lapply(itemM, unique)
itemM <- as(itemM, "transactions"); itemM
WC <- as(WC, "transactions"); WC
itemFrequencyPlot(itemM,support=.20)
sizes<-size(itemM)
```

```
sizes

size.labels<-as.numeric(levels(as.factor(sizes)))

itemM.subset.2<-subset(itemM,sizes==2)

inspect(itemM.subset.2)

HEBdatasup <- itemFrequency(itemM, type= "relative")

M = mean(HEBdatasup)

rules_class1<- apriori(itemM,

           parameter=list(support=.2,

                     confidence=.8, maxlen=2, target="rules"))

rules_class1

inspect(rules_class1)

HEBdatasup <- itemFrequency(itemM, type= "relative")

sort(head(HEBdatasup, 25), decreasing = TRUE)

highest.lift_class1 <- sort(rules_class1, by = "lift", na.last=NA, decreasing = TRUE)

inspect(head(highest.lift_class1, 25))

highest.conf_class1 <- sort(rules_class1, by = "confidence", na.last=NA, decreasing =

TRUE)

inspect(head(highest.conf_class1, 25))

highest.sup_class1 <- sort(rules_class1, by = "support", na.last=NA, decreasing = TRUE)

inspect(head(highest.sup_class1, 25))

association_rules_class1<-as(rules_class1, "data.frame");

rules_highest_lift_class1 <- as(highest.lift_class1, "data.frame");

write.csv(association_rules_class1,"association_rules_class1.csv")
write.csv(rules_highest_lift_class1, "rules_highest_lift_class1.csv")

## Clean Association Rules output

before1 <- rules_highest_lift_class1

out1 <- strsplit(as.character(before1$rules),'=>')

out1 <- data.frame(t(sapply(out1, `[`)))

after1 <- with(before1, data.frame(support = support, confidence = confidence, lift = lift))

after1 <- cbind(out1,after1)

names(after1)[1:2] <- c("LHS", "RHS")

after1$LHS<-gsub('.{2}$', '', after1$LHS)

after1$LHS<-gsub('^.', '', after1$LHS)

after1$RHS<-gsub('.{1}$', '', after1$RHS)

after1$RHS<-gsub('^.{2}', '', after1$RHS)

write.csv(after1, "rules_highest_lift_classs1_cleaned.csv")

## Association rules mining for class #2

install.package("arules")

library("arules")

HEBdata <- weight_filter_2

HEBdata<-transform(HEBdata, ASGN=as.factor(ASGN))

HEBdata<-transform(HEBdata, Product=as.factor(Product))

HEBdata<-transform(HEBdata, Aisle=as.factor(Aisle))

HEBdata<-transform(HEBdata, Location=as.factor(Location))

HEBdata<-transform(HEBdata, Description=as.factor(ItemDescription))

HEBdata<-transform(HEBdata, Weight=as.numeric(Weight))
```

```
length(levels(HEBdata$ASGN))

length(levels(HEBdata$Product))

quantile(HEBdata$Weight)

WC <- split(x=HEBdata[,"Description"], f=HEBdata$ASGN)

itemM <- split(x=HEBdata[,"Product"], f=HEBdata$ASGN)

WC <- lapply(WC, unique)

itemM <- lapply(itemM, unique)

itemM <- as(itemM, "transactions"); itemM

WC <- as(WC, "transactions"); WC

itemFrequencyPlot(itemM,support=.20)

sizes<-size(itemM)

sizes

size.labels<-as.numeric(levels(as.factor(sizes)))

itemM.subset.2<-subset(itemM,sizes==2)

inspect(itemM.subset.2)

HEBdatasup <- itemFrequency(itemM, type= "relative")

M = mean(HEBdatasup)

rules_class2<- apriori(itemM,

            parameter=list(support=.2,

                        confidence=.8, maxlen=2, target="rules"))

rules_class2

inspect(rules_class2)

HEBdatasup <- itemFrequency(itemM, type= "relative")
```

```
sort(head(HEBdatasup, 25), decreasing = TRUE)

highest.lift_class2 <- sort(rules_class2, by = "lift", na.last=NA, decreasing = TRUE)

inspect(head(highest.lift_class2, 25))

highest.conf_class2 <- sort(rules_class2, by = "confidence", na.last=NA, decreasing =
TRUE)

inspect(head(highest.conf_class2, 25))

highest.sup_class2 <- sort(rules_class2, by = "support", na.last=NA, decreasing = TRUE)

inspect(head(highest.sup_class2, 25))

association_rules_class2<-as(rules_class2, "data.frame");

rules_highest_lift_class2 <- as(highest.lift_class2, "data.frame");

write.csv(association_rules_class2,"association_rules_class2.csv")

write.csv(rules_highest_lift_class2, "rules_highest_lift_class2.csv")

## Clean Association Rules output

before2 <- rules_highest_lift_class2

out2 <- strsplit(as.character(before2$rules),'=>')

out2 <- data.frame(t(sapply(out2, `[`)))

after2 <- with(before2, data.frame(support = support, confidence = confidence, lift = lift))

after2 <- cbind(out2,after2)

names(after2)[1:2] <- c("LHS", "RHS")

after2$LHS<-gsub('.{2}$', '', after2$LHS)

after2$LHS<-gsub('^.', '', after2$LHS)

after2$RHS<-gsub('.{1}$', '', after2$RHS)

after2$RHS<-gsub('^.{2}', '', after2$RHS)
```

```
write.csv(after2, "rules_highest_lift_classs2_cleaned.csv")

## Association rules mining for class #3

install.package("arules")

library("arules")

HEBdata <- weight_filter_3

HEBdata<-transform(HEBdata, ASGN=as.factor(ASGN))

HEBdata<-transform(HEBdata, Product=as.factor(Product))

HEBdata<-transform(HEBdata, Aisle=as.factor(Aisle))

HEBdata<-transform(HEBdata, Location=as.factor(Location))

HEBdata<-transform(HEBdata, Description=as.factor(ItemDescription))

HEBdata<-transform(HEBdata, Weight=as.numeric(Weight))

length(levels(HEBdata$ASGN))

length(levels(HEBdata$Product))

quantile(HEBdata$Weight)

WC <- split(x=HEBdata[,"Description"], f=HEBdata$ASGN)

itemM <- split(x=HEBdata[,"Product"], f=HEBdata$ASGN)

WC <- lapply(WC, unique)

itemM <- lapply(itemM, unique)

itemM <- as(itemM, "transactions"); itemM

WC <- as(WC, "transactions"); WC

itemFrequencyPlot(itemM,support=.20)

sizes<-size(itemM)

sizes
```

```
size.labels<-as.numeric(levels(as.factor(sizes)))
itemM.subset.2<-subset(itemM,sizes==2)
inspect(itemM.subset.2)
HEBdatasup <- itemFrequency(itemM, type= "relative")
M = mean(HEBdatasup)
rules_class3<- apriori(itemM,
        parameter=list(support=.2,
                confidence=.8, maxlen=2, target="rules"))
rules_class3
inspect(rules_class3)
HEBdatasup <- itemFrequency(itemM, type= "relative")
sort(head(HEBdatasup, 25), decreasing = TRUE)
highest.lift_class3 <- sort(rules_class3, by = "lift", na.last=NA, decreasing = TRUE)
inspect(head(highest.lift_class3, 25))
highest.conf_class3 <- sort(rules_class3, by = "confidence", na.last=NA, decreasing =
TRUE)
inspect(head(highest.conf_class3, 25))
highest.sup_class3 <- sort(rules_class3, by = "support", na.last=NA, decreasing = TRUE)
inspect(head(highest.sup_class3, 25))
association_rules_class3<-as(rules_class3, "data.frame");
rules_highest_lift_class3 <- as(highest.lift_class3, "data.frame");
write.csv(association_rules_class3,"association_rules_class3.csv")
write.csv(rules_highest_lift_class3, "rules_highest_lift_class3.csv")
```

```
## Clean Association Rules output

before3 <- rules_highest_lift_class3

out3 <- strsplit(as.character(before3$rules),'=>')

out3 <- data.frame(t(sapply(out3, `[`)))

after3 <- with(before3, data.frame(support = support, confidence = confidence, lift = lift))

after3 <- cbind(out3,after3)

names(after3)[1:2] <- c("LHS", "RHS")

after3$LHS<-gsub('.{2}$', '', after3$LHS)

after3$LHS<-gsub('^.', '', after3$LHS)

after3$RHS<-gsub('.{1}$', '', after3$RHS)

after3$RHS<-gsub('^.{2}', '', after3$RHS)

write.csv(after3, "rules_highest_lift_classs3_cleaned.csv")
```

APPENDIX 5: AUTOMOD CODE FOR CREATING SIMULATION LOGICS (PARTIAL)

```
SFileBegin    name init.m
begin model initialization function
        call F_Location_Init()
        call F_Order_Init()
create 39 load of type Order_load to PMasterLoad
        return true
end
begin PMasterLoad arriving procedure
        inc V_Order_ctr by 1
        set Attr_Order_ID to V_Order_ctr
        call SReceiveOrder
        call SPickOrder
        call SDeliverOrder
        print ac to message
end
SFileBegin    name veh.m
begin pm vehicle initialization function
        increment V_Count by 1
        set theVehicle A_index = V_Count
        set theVehicle A_Home = pm.cp_home
        dispatch theVehicle to pm.cp_home
        return true
end
SFileBegin    title "idle, receive,pick,deliver"
        name subs.m
begin pm idle procedure
        dispatch this vehicle to pm.cp_home
        wait to be ordered on OLVehicle
end
begin SReceiveOrder procedure
        move into QInitial
        move into pm.cp_home
end
begin SPickOrder procedure
        set Attr_Time2 to ac
        set Vi to 1
        print this load to message
        print "Order received:" ac to message
        print this load vehicle to message
        while Vi <= V_Number_Item_Per_Ord(Attr_Order_ID) do
        begin
                travel to V_Location(V_Order(Attr_Order_ID,Vi))
                inc Vi by 1
                inc V_Number_Done by 1
        end
end
begin SDeliverOrder procedure
        tabulate ac - Attr_Time2 in T_PickTime
        print "finish time of picking up the last SKU in an order" ac to message
        tabulate V_Number_Done in T_Throughput
        travel to pm.cp_home
        print "time back to Home for another order" ac to message
end
```

REFERENCES

[1] Koster, R. D., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. European Journal of Operational Research,182(2), 481-501. doi:10.1016/j.ejor.2006.07.009

[2] Tompkins, J.A., J.A. White, Y.A. Bozer, E.H. Frazelle, J.M.A. Tanchoco and J. Trevino (1996). Facilities Planning, 2nd ed., New York: Wiley & Sons Inc.

[3] Ming-Huang Chiang, D., Lin, C.-P. and Chen, M.-C. (2012), Data mining based storage assignment heuristics for travel distance reduction. Expert Systems, 31: 81–90. doi:10.1111/exsy.12006

[4] Chen, M., Huang, C., Chen, K., & Wu, H. (2005). Aggregation of orders in distribution centers using data mining. Expert Systems with Applications,28(3), 453-460. doi:10.1016/j.eswa.2004.12.006

[5] Chen, M., & Wu, H. (2005). An association-based clustering approach to order batching considering customer demand patterns. Omega,33(4), 333-343. doi:10.1016/j.omega.2004.05.003

[6] Diaper-beer syndrome. (1998, April 06). Retrieved July 19, 2017, from https://www.forbes.com/forbes/1998/0406/6107128a.html

[7] ELA/AT Kearney (2004). Excellence in Logistics 2004. ELA, Brussels.

[8] Vaughan, T. S. (1999). The effect of warehouse cross aisles on order picking efficiency. International Journal of Production Research,37(4), 881-897. doi:10.1080/002075499191580

[9] Petersen, C.G. (2000). An evaluation of order picking policies for mail order companies. Production and Operations Management 9 (4), 319–335.

[10] Ratliff, H. D., & Rosenthal, A. S. (1983). Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. Operations Research,31(3), 507-521. doi:10.1287/opre.31.3.507

[11] Hall, R. W. (1993). Distance Approximations For Routing Manual Pickers In A Warehouse. IIE Transactions,25(4), 76-87. doi:10.1080/07408179308964306

[12] Petersen, C.G. (1997). An evaluation of order picking routing policies. International Journal of Operations & Production Management 17 (11), 1098–1111.

[13] Roodbergen, K.J. (2001). Layout and routing methods for warehouses. Ph.D. thesis, RSM Erasmus University, the Netherlands.

[14] Petersen, C. G., Aase, G. R., & Heiser, D. R. (2004). Improving order- picking performance through the implementation of class- based storage. International Journal of Physical Distribution & Logistics Management,34(7), 534-544. doi:10.1108/09600030410552230

[15] Frazelle, E.A., & Sharp, G.P.(1989). Correlated assignment strategy can improve order-picking operation. Industrial Engineering, 4, 33-37.

[16] Hsieh, L.F., & Huang, C.L. (2007). Optimal order picking planning for distribution center with cross aisle. Proceedings of the 7th International Conference on Optimization: Techniques and Applications (ICOTA7), Kobe, Japan.

[17] Bassan, Y., & Roll, Y., & Rosenblatt, M.J. (1980). Internal layout design of a warehouse. AIIE Transactions 12 (4), 317–322.

[18] Roll, Y., & Rosenblatt, M.J. (1983). Random versus grouped storage policies and their effect on warehouse capacity. Material Flow 1, 199–205.

[19] Caron, F., Marchet, G., & Perego, A. (2000). Optimal layout in low-level picker-to-part systems. International Journal of Production Research,38(1), 101-117. doi:10.1080/002075400189608

[20] R. (M.) B. M. De Koster, Le-Duc, T., & Yugang, Y. (2008). Optimal storage rack design for a 3-dimensional compact AS/RS. International Journal of Production Research,46(6), 1495-1514. doi:10.1080/00207540600957795

[21] Petersen, C. G. (2002). Considerations in order picking zone configuration. International Journal of Operations & Production Management,22(7), 793-805. doi:10.1108/01443570210433553

[22] Choe, K., & Sharp, G.P. (1991). Small parts order picking: design and operation.

[23] Hausman, W. H., & Schwarz, L. B., & Graves, S. C. (1976). Optimal Storage Assignment in Automatic Warehousing Systems. Management Science,22(6), 629-638. doi:10.1287/mnsc.22.6.629

[24] Heskett, J. L. (1963). Cube-per-order index-a key to warehouse stock location. Transportation and distribution Management, 3(1), 27-31.

[25] Kallina, C., & Lynn, J. (1976). Application of the cube-per-order index rule for stock location in a distribution warehouse. Interfaces 7 (1), 37–46.

[26] Malmborg, C.J., & Bhaskaran, K. (1990). A revised proof of optimality for the cube-per-order index rule for stored item location. Applied Mathematical Modelling (14), 87–95.

[27] Malmborg, C.J. (1995). Optimization of Cubic-per-Order Index layouts with zoning constraints. International Journal of Production Research 33 (2), 465–482.

[28] Caron, F., & Marchet, G., & Perego, A. (1998). Routing policies and COI-based storage policies in picker-to-part systems. International Journal of Production Research 36 (3), 713–732.

[29] Petersen, C. G., & Aase, G. (2004). A comparison of picking, storage, and routing policies in manual order picking. International Journal of Production Economics,92(1), 11-19. doi:10.1016/j.ijpe.2003.09.006

[30] Yang, M. (1988, May 01). Analysis and optimization of class-based dedicated storage systems. Retrieved July 19, 2017, from http://hdl.handle.net/1853/21730

[31] Van den Berg, J.P., & Gademann, A.J.R.N. (2000). Simulation study of an automated storage/retrieval system. International Journal of Production Research 38, 1339–1356.

[32] Tompkins, J. A., & Smith, J. D. (1998). The Warehouse management handbook. Raleigh, NC: Tompkins Press.

[33] Hsu, C., Chen, K., & Chen, M. (2005). Batching orders in warehouses by minimizing travel distance with genetic algorithms. Computers in Industry,56(2), 169-178. doi:10.1016/j.compind.2004.06.001

[34] Petersen, C. G., & Schmenner, R. W. (1999). An Evaluation of Routing and Volume-based Storage Policies in an Order Picking Operation. Decision Sciences,30(2), 481-501. doi:10.1111/j.1540-5915.1999.tb01619.x

[35] Chen, L., Langevin, A., & Riopel, D. (2011). A tabu search algorithm for the relocation problem in a warehousing system. International Journal of Production Economics,129(1), 147-156. doi:10.1016/j.ijpe.2010.09.012

[36] Chuang, Y., Lee, H., & Lai, Y. (2012). Item-associated cluster assignment model on storage allocation problems. Computers & Industrial Engineering,63(4), 1171-1177. doi:10.1016/j.cie.2012.06.021

[37] Mantel, R. J., Schuur, P. C., & Heragu, S. S. (2007). Order oriented slotting: a new assignment strategy for warehouses. European J. of Industrial Engineering,1(3), 301. doi:10.1504/ejie.2007.014689

[38] Diaz, R. (2015). Using dynamic demand information and zoning for the storage of non-uniform density stock keeping units. International Journal of Production Research,54(8), 2487-2498. doi:10.1080/00207543.2015.1106605

[39] Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD 93. doi:10.1145/170035.170072

[40] Han, J., Kamber, M., & Pei, J. (2011). Data mining: Concepts and techniques concepts and techniques. San Francisco: Morgan Kaufmann In.

[41] Diaz, R. (2010). Using optmization coupled with simulation to construct layout solutions. Proceedings of the 2010 Spring Simulation Multiconference on - SpringSim 10. doi:10.1145/1878537.1878617

[42] Chan, H., Pang, A., & Li, K. (1970, January 01). Association rule based approach for improving operation efficiency in a randomized warehouse. Retrieved July 19, 2017, from http://hdl.handle.net/10397/4450

[43] Mining Association Rules and Frequent Itemsets [R package arules version 1.5-2]. (n.d.). Retrieved July 19, 2017, from https://cran.r-project.org/web/packages/arules/index.html