

**DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES**

**EVALUATION OF ORDER PICKING SYSTEMS
USING SIMULATION**

**by
Gökçeçişek TUNA TAŞOĞLU**

**January, 2013
İZMİR**

EVALUATION OF ORDER PICKING SYSTEMS USING SIMULATION

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
Master of Science in Industrial Engineering**

**by
Gökçeçişek TUNA TAŞOĞLU**

January, 2013

İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled “**EVALUATION OF ORDER PICKING SYSTEMS USING SIMULATION**” completed by **GÖKÇEÇİÇEK TUNA TAŞOĞLU** under supervision of **ASST. PROF. DR. GONCA TUNÇEL MEMİŞ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



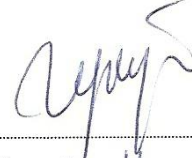
Yrd. Doç. Dr. Gonca TUNÇEL MEMİŞ

Supervisor



Yrd. Doç. Dr. Gökay YILDIZ

(Jury Member)



Yrd. Doç. Dr. Umay KOCER

(Jury Member)



Prof. Dr. Mustafa SABUNCU

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

This thesis would not have been completed without the support of many people.

First I would like to thank my supervisor Asst. Prof. Dr. Gonca TUNÇEL MEMİŞ for she guided me throughout the whole study patiently and helped me to complete the research and write this thesis and I wish to express my gratitude to Asst. Prof. Dr Gökalp YILDIZ who was abundantly helpful and offered invaluable assistance, support and guidance.

I also take this opportunity to thank the staff in the Department of Industrial Engineering at Celal Bayar University for their help throughout my M.Sc. study.

Finally I am forever indebted to my parents, my brother and my husband for their understanding, endless patience and encouragement when it was most required.

Gökçeçiçek TUNA TAŞOĞLU

EVALUATION OF ORDER PICKING SYSTEMS USING SIMULATION

ABSTRACT

Order picking activities play a critical role in supply chain management in terms of both production systems (supplying components to assembly operations) and distribution operations (meeting customer demands). Trends in customer orders reveal that orders are transformed from few-and-large orders to many-and-small ones. On the other hand, lead times of customer orders get consistently shorter. Because of these changes, companies need to adopt an effective and flexible order picking system in order to remain competitive in the market. Order picking affects both overall logistic operations and service level provided to customers. Additionally, order picking process constitutes more than half of the total warehousing cost. For these reasons, it is crucial for companies to design and perform an effective order picking process.

The aim of this study is evaluating and improving of the order picking system so as to minimize the order retrieval time while increasing the picking efficiency. Order retrieval time can be defined as the time elapsed for the process of retrieving products from storage area to meet a specific customer demand. Besides the critical factors such as storage assignment decisions and routing methods, replenishment problem of the storage areas, which is rarely addressed in the previous studies, has been taken into consideration in this study. Replenishment of the empty storage locations has been conducted by using the (S, s) inventory policy. Thus, the order picking system was modeled as a dynamic system.

A hypothetical distribution warehouse, based on the real life warehouse of a company specialized in production of fasteners, has been studied in order to improve the order picking performance. Alternative combinations of routing and storage policies have been developed. Moreover, different simulation models of the order picking process were constructed. In these models, proposed alternative storage and

routing policies were operated. According to the simulation results, the storage policy and routing policy combination which provides the shortest order retrieval time is determined. Finally, using simulation results, some statistical analysis methods have been implemented.

Keywords: Order picking, warehouse management, design of order picking system, order replenishment, (S, s) inventory policy.

SİPARİŞ TOPLAMA SİSTEMLERİNİN SİMULASYON KULLANARAK DEĞERLENDİRİLMESİ

ÖZ

Sipariş toplama faaliyetleri, tedarik zinciri yönetiminde, hem üretim sistemleri açısından (montaj istasyonlarına alt parçaların tedarik edilmesi), hem de dağıtım işlemleri açısından (müşteri taleplerinin karşılanması) kritik rol oynamaktadır. Müşteri siparişlerindeki eğilimler, az sayıda ve yüksek miktarlarda siparişlerin çok sayıda ve düşük miktarlarda siparişlere dönüştüğünü göstermektedir. Diğer yandan, talep edilen sipariş teslim süreleri ise her geçen gün kısalmaktadır. Bu değişimler, işletmelerin piyasada rekabet edebilmeleri için etkin ve esnek bir sipariş toplama sistemi benimsemelerini gerektirmektedir.

Sipariş toplama süreci, tüm lojistik operasyonlarını ve müşteriye sağlanan hizmet seviyesini büyük ölçüde etkilemektedir. Ayrıca, sipariş toplama süreci toplam depolama maliyetlerinin yarısından fazlasını oluşturmaktadır. Bu nedenle, sipariş toplama faaliyetlerinin en etkin şekilde gerçekleştirilmesi işletmeler için büyük önem taşımaktadır.

Bu çalışmanın amacı, sipariş toplama süresini kısaltarak, sipariş toplama etkinliğini arttırmaya yönelik değişiklikler için sipariş toplama sistemini değerlendirmek ve geliştirmektir. Sipariş toplama süresi, ürünlerin depolama alanlarından belirli bir müşteri talebini karşılamak amacıyla toplanması süreci için geçen zamandır. Bu çalışmada, ürünlerin depolama alanlarına atanma kararları ve rotalama metotları gibi kritik faktörlerin yanı sıra, daha önce gerçekleştirilmiş çalışmalarda sıkça rastlanmayan depolama alanlarının ikmal problemi dikkate alınmıştır. Boşalan rafların yeniden doldurulması kararında, (S, s) envanter politikası uygulanmıştır. Böylece, sipariş toplama sistemi dinamik olarak modellenmiştir.

Sipariş toplama performansını geliştirmek için, bağlantı elemanları üreten bir firmanın ambarı temel alınarak oluşturulmuş hipotetik bir ambar üzerinde

alıřılmıřtır. Ambara ait farklı benzetim modelleri oluřturulmuř, depolama ve rotalama politikalarının alternatif kombinasyonları geliřtirilerek bu benzetim modellerinde kullanılmıřtır. Elde edilen benzetim sonularına gre, en kk sipariř toplama sresini veren depolama ve rotalama politikası kombinasyonu belirlenmiřtir. Son olarak, benzetim sonuları zerinde bazı istatistiksel analiz metotları uygulanmıřtır.

Anahtar szckler: Sipariř toplama, depo ynetimi, sipariř toplama sistemi tasarımı, depolama alanlarının ikmali, (S, s) envanter politikası.

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ	vi
CHAPTER ONE – INTRODUCTION	1
1.1 Warehouse Management	1
1.2 Order Picking in Warehouse Management	5
CHAPTER TWO –CLASSIFICATION, DESIGN AND EVALUATION OF ORDER PICKING SYSTEMS	7
2.1 Decisions that Affect the Classification of Order Picking Systems	7
2.1.1 Order Picking Strategies	7
2.1.2 Order Picking Methods	8
2.1.3 Material Handling Equipment	8
2.2 Classification of Order Picking Systems	9
2.2.1 Picker-to-Parts Systems	10
2.2.2 Pick-to-Box/Pick-and-Pass Systems	11
2.2.3 Pick-and-Sort Systems	12
2.2.4 Parts-to-picker Systems	12
2.2.5 Completely Automated Systems	13
2.3 Design and Evaluation of the Order Picking Systems	13
2.3.1 Layout Design	14
2.3.2 Storage Assignment Policies	15
2.3.2.1 Random Storage	16
2.3.2.2 The Nearest Empty Location Storage	16

2.3.2.3 Dedicated Storage	16
2.3.2.4 Full Turnover Storage	17
2.3.2.5 Class Based Storage	18
2.3.3 Order Picking Route Determination	19
2.3.3.1 S-Shaped/Transversal Heuristic.....	20
2.3.3.2 Return Heuristic	20
2.3.3.3 Mid-Point Heuristic	21
2.3.3.4 Largest Gap Heuristic	22
2.3.3.5 Combined (Composite) Heuristic	23
2.3.4 Zoning.....	24
2.3.5 Batching	25
2.3.6 Order Accumulation/Sorting	25
CHAPTER THREE– LITERATURE REVIEW	28
CHAPTER FOUR –EVALUATION OF ORDER PICKING SYSTEMS USING SIMULATION	35
4.1 Problem Statement	35
4.2 Solution Methodology	36
4.2.1 Implementation of the Simulation Model.....	37
4.3 Output Analysis.....	46
4.3.1 Verification and Validation of Simulation Model.....	48
4.3.2 Determining the Length of Warm-up Period.....	49
4.3.3 Confidence Interval Estimation	52
4.3.4 All Pairwise Comparisons	56
CHAPTER FIVE-RESULTS AND FUTURE RESEARCH.....	63
REFERENCES.....	65
APPENDICIES.....	71

CHAPTER ONE

INTRODUCTION

1.1 Warehouse Management

Warehouses are usually used for keeping raw materials, components, semi-finished and finished products, between production area and customers. In addition, warehouses can be used for keeping products until they are distributed to the customers.

Different types of warehouses which perform different purposes exist in the supply chain. A warehouse can be included within more than one category at the same time. Here are the main types of the warehouses:

1. *Raw material and component warehouses:* These are the warehouses which store the raw materials and components used for the production in a production facility (factory, workshop etc.). These warehouses are usually located in the same building with the production facility or they may be neighbors.
2. *Work-in-process warehouses:* These are the warehouses which store the semi-finished products. They generally located near the production lines or production cells.
3. *Finished goods warehouses:* Finished products are stored in these warehouses.
4. *Distribution warehouses and distribution centers:* In these types of warehouses, numerous products that come from the different suppliers are consolidated with respect to order contents. Afterwards, orders are distributed to the customers.
5. *Value-added service warehouses:* Various value-added operations are performed to the stored products in these warehouses.
6. *Local warehouses:* Local warehouses provide service to the regions with the high customer demand. These warehouses are located close to the customers

in order to reduce order lead time, so customer demands are met in shorter time.

7. *Customs bonded warehouse:* These warehouses are operated with respect to the permissions given by the undersecretaries of customs. They depend on a custom administration. The owner of the warehouse should be a legal person or a corporation. Products that are going to be exported are stored in the warehouse. These warehouses become very important when international trade is considered.

Warehouse management involves the decisions such as where to store the products, in which conditions the products should be stored and controlling of considerable warehousing costs and acceleration of the products flow. Location of the warehousing in the general scope of the logistics is depicted in Figure 1.1.

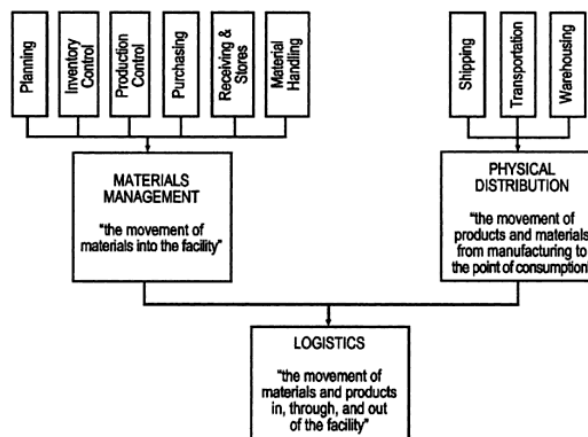


Figure 1.1 General scope of the logistics

In warehouse management, product flow is divided into three basic processes; receiving, storage and shipping (Rouwenhorst et al., 2000). On the other hand, warehousing operations generally include the following processes:

- Receiving,
- Put away,
- Storage,
- Replenishment,

- Order Picking,
- Accumulation/ Sortation and Packaging,
- Shipment.

Receiving is the first process starting with the arrival of products. Products arrive by a truck or an internal transporter used in the production area. In this process, products are checked for signs of deterioration and they are transformed into another storage module if it is necessary. Finally, they wait for the transportation to the next process.

In the storage process, products are placed in the storage locations. In order to accelerate the order picking process, most distribution centers store stock keeping units in two different storage areas; forward area and reserve area. Reserve area is the area where the products are stored for longer time periods in the most effective way, while forward area is the area where products are stored in smaller amounts for easy and quick retrieval by the order pickers. In other words, the forward area is used for efficient order picking, whereas the reserve area is used for efficiently storing the large amount of products used to replenish the forward area. For example, in the reserve area pallet racks are used for storage, on the other hand, in the forward area products are stored on the shelves. Size of the forward area is restricted to decrease mean order retrieval time and accelerate the order picking process. Determining the sizes of the forward area and reserve area and the amount of the products allocated to these respective areas is an important decision. The transfer of products from the reserve area to the forward area is called as replenishment. Determination of the frequency of replenishment is another important decision. These problems are gathered under the *forward-reserve problem*.

In warehouse management, different product flows can be observed among the receiving, storage and shipping stages. Heragu et al. (2005) introduced four different types of flows that may occur while storing the products. These flows vary with respect to the aim of warehousing activity. In Figure 1.2, the typical product flow in warehouse management is depicted.

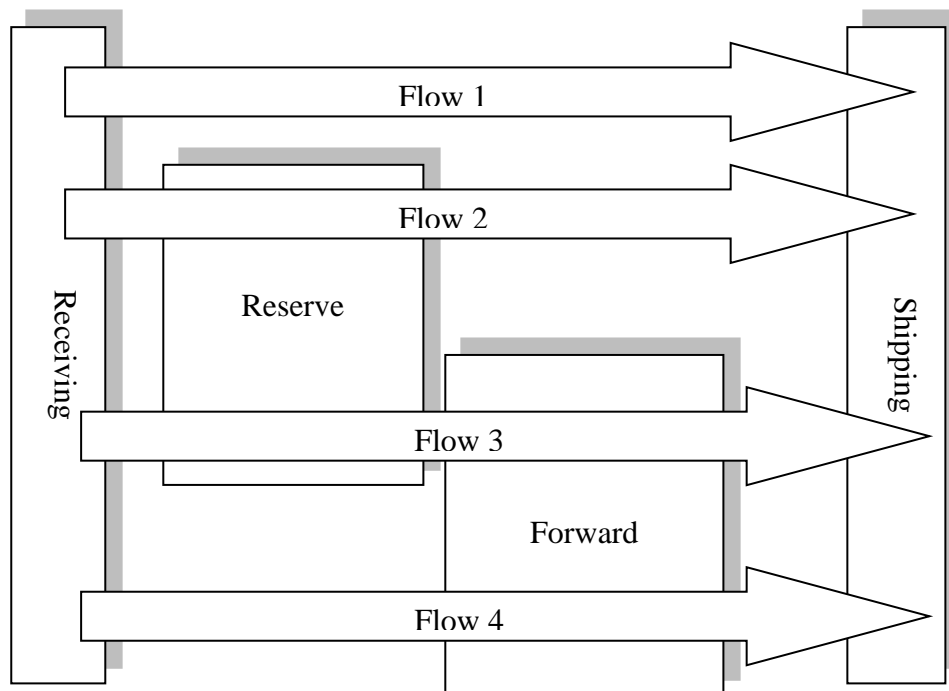


Figure 1.2 Typical product flows in a warehouse (Heragu et al., 2005)

Flow 1 is the cross-docking operation. Upon receipt, product items are either put into a staging area for a short period and then moved to the shipping area, or directly moved to the shipping area.

Flow 2 is a typical warehouse operation. Products are stored in a reserve area and order-picking operation is performed as required. It is assumed that, typically, only those items that remain in the warehouse for relatively extended periods and shipped as is (or with minimal value added operations) will be allocated to the reserve storage area.

Flow 3 is also a typical warehouse operation. Products are first stored in the reserve area typically in pallet loads, broken into smaller loads (cartons or cases) and then moved to the forward area for fast order picking, order consolidation or performing value added operations.

Flow 4 can be thought of as another form of cross-docking operation. Products are received and then are directly put into forward area to perform the order

consolidation. This type of operation is usually seen in the supplier warehouses or when there is a need to consolidate large orders (Heragu et al, 2005).

1.2 Order Picking Process in Warehouse Management

Order picking, which corresponds to picking goods from warehouses to meet customer demands, is considered to be the most important warehousing activity. The most common objective to improve the order picking process, beside the effective use of the limited resources such as labor, machine and money, is to maximize service level. The basic connection between the order picking process and the service level stems from the need for creating the order quickly and having it ready for delivery as soon as possible. Service level consists of a combination of order accuracy, order unity and variability in mean order retrieval time. Figure 1.3 shows components of the order retrieval time. Almost fifty percent of the order retrieval time is consumed by travel time. According to Bartholdi and Hackman (2005), “Travel time is waste, it causes labor cost and it is a non value added activity, so it is the first candidate component to improve in an order picking system.” (Bartholdi and Hackman, 2005).

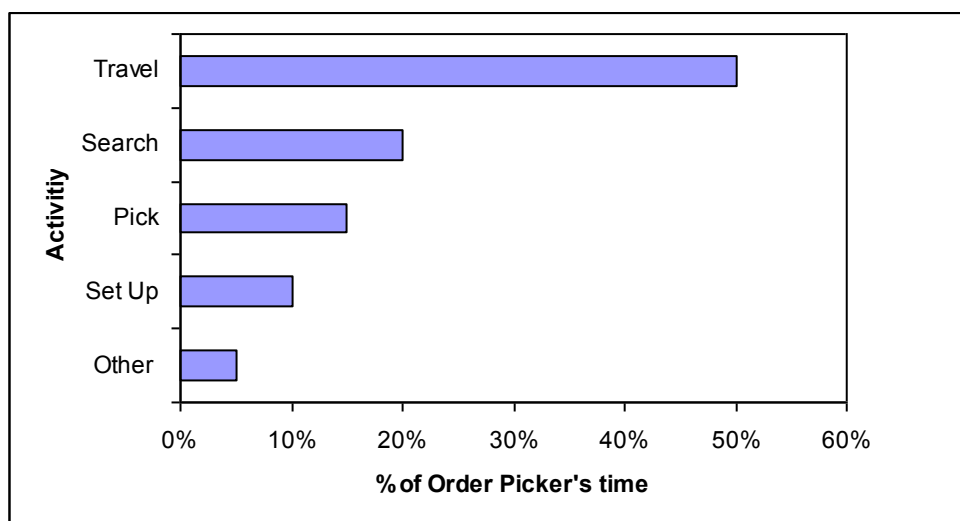


Figure 1.3 Components of Order Picking Time (Tompkins et al, 2003)

In addition to minimizing the travel time, following objectives are considered while designing and controlling the warehouse:

- Minimize the order retrieval time
- Minimize the overall time to complete a batch of orders
- Maximize the use of space in the warehouse facility
- Maximize the use of labor
- Maximize the use of equipment
- Maximize the accessibility to all products in the warehouse.

The remainder of the thesis is organized as follows. Chapter 2 provides the basic definitions for the classification of the order picking systems. Additionally, fundamentals of the design, control and evaluation of the order picking systems are presented. Factors that affect the performance of the order picking are described in detail. In Chapter 3, related literature is reviewed. The development of the simulation model representing a hypothetical order picking system is proposed in Chapter 4, along with the assumptions of the model. Statistical analysis of the simulation results are also considered in Chapter 4. In Chapter 5, results and research directions are exhibited.

CHAPTER TWO

CLASSIFICATION, DESIGN AND EVALUATION OF THE ORDER PICKING SYSTEMS

2.1 Decisions That Affect the Classification of the Order Picking Systems

Order picking systems are divided into various classes according to the used equipment and operating policies. For this classification, there are four main decisions that are crucial. These decisions are; (I) who collects the products (human / machine), (ii) what is moving in the storage area? (collectors / parts), (iii) what are the current material handling systems used between the existing order picking zones? (conveyors, etc.) and (iv) which order picking policy is implemented in the warehouse? (Dallari et al., 2009). In addition to these decisions, following strategies have effect on the order picking system classification:

- Order Picking Strategy
- Order Picking Method
- Material Handling Equipment

2.1.1 Order Picking Strategy

An order picking strategy defines the manner in which the order pickers behave in the aisles of a storage area to pick the products in an order list. Basic order picking strategies are described as follows:

- *discrete picking*: order picker picks all the products in a single order during a pick-tour;
- *batch picking*: several orders are batched (or grouped) together and a order picker retrieves all the products in a proposed batch;
- *zone picking*: each picker is assigned to a particular region of the storage area and is responsible for picking the products in that region only.

2.1.2 Order Picking Method

Depending on whether or not humans are involved in the system, there are three methods in which order picking can be accomplished; namely, *manual*, *semi-automated*, and *automated*. A manual (or picker-to-parts) Order Picking System (OPS) is one in which the order pickers travel to the point where the item to be picked is located. A semi-automated (or parts-to-picker) OPS is one in which the items to be picked are brought to a stationary picker through mechanical means (e.g., carousel, vertical lift module, etc.). An automated OPS has the potential of picking orders without any human intervention (e.g., A-frame).

2.1.3 Material Handling Equipment

In an order picking system, material handling equipment can be used either to assist in manual, semi-automated, or automated picking of items. A variety of material handling equipment is employed in a modern Distribution Centre (DC) to increase the productivity of order picking system. These include totes, carts, conveyors, trucks (counter-balanced (CB) lift, order-picker, reach, etc.), horizontal and vertical carousels, vertical lift modules (VLMs), A-frame dispenser, end-of-aisle mini-load automated storage and retrieval systems (AS/RS), etc. Some of these equipments are illustrated in Figure 2.1.

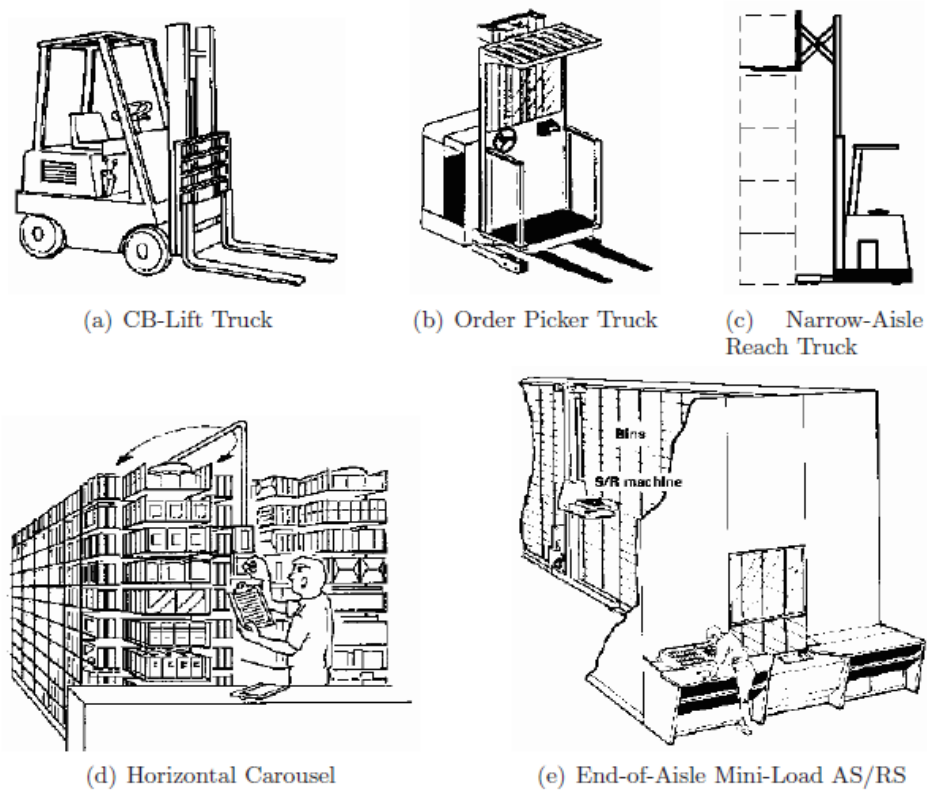


Figure 2.1 Material handling equipment employed within an order picking system (MHIA, 2006)

2.2 Classification of Order Picking Systems

With respect to decisions mentioned in the previous sections, Dallari et al., (2009) classified the order picking systems as follows:

- a) Picker-to-parts
- b) Pick-to-box/ Pick-and-pass
- c) Pick-and-sort
- d) Parts-to-picker
- e) Completely automated picking

As shown in Figure 2.2, in the growing variety from “Picker-to-parts” order picking systems to “completely automated picking”, there is an increase in the level of automation.

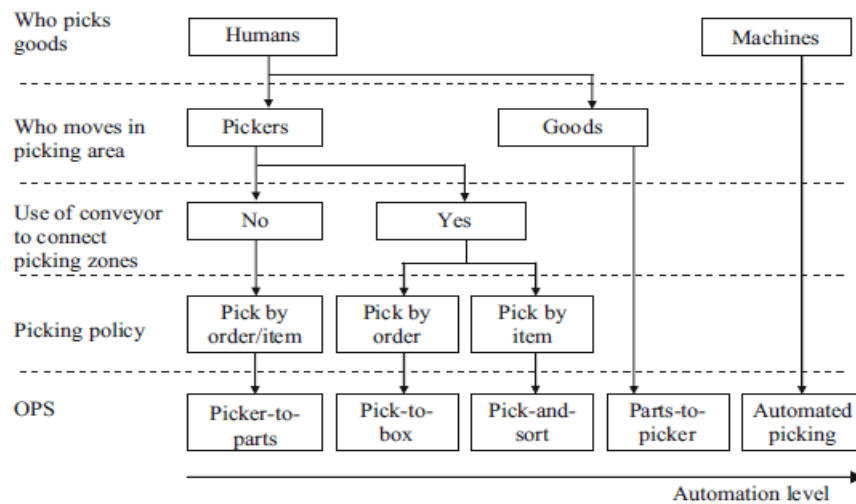


Figure 2.2 Classification order picking systems (Dallari et al., 2009)

2.2.1 Picker-to-Parts Order Picking Systems

The most commonly used order picking system, “picker-to-parts” order picking, can be considered as the basis of warehousing activities. In such systems, order picker travels on foot or by driving a transporter through the aisles of the warehouse and picks the products to complete an individual customer order or a batch of different customer orders. Also, if order picker completes a batch of orders, it sorts the items according to the customer orders which they belong to. This situation especially occurs in sort-while-pick systems.

Picker-to-parts order picking systems have been divided into two groups; low level order picking systems and high level order picking systems (Caron et al., 2000). In low level systems, order picker collects the products from the boxes, bins or shelves while moving along the aisles. Low level order picking systems are generally used in warehouses because of their low initial cost, easy installation, easy reformation and low maintenance cost.

On the other hand, in high level order picking systems, order picker reaches the overhead shelves and storage areas by using automated vehicles such as crane or lifting truck. In other words, high level picking systems employ high storage

locations. Shelves, storage bins or cabinets can be located as high as floor loading, weight limits and ceiling heights will permit. When it comes to use of crane, it automatically stops in front of the proposed picking bay and waits for the order picker to perform the pick. So, these systems are called also as man-on-board systems. Compared to a low level system, a man on board system has higher installation and maintenance cost and lower reformation ability.

In a “picker-to-parts” system, supplementary developments can be accomplished with respect to routing algorithms, product assignment policies and paperless activities using radio frequency or voice oriented technologies.

2.2.2 Pick-to-Box Order Picking Systems

When pick-to-box or pick-and-pass order picking systems are considered, picking area is divided into different regions. Each region is assigned to one or more order pickers. All of the order picking regions are connected to each other with a conveyor. There are boxes moving on the conveyor. Boxes visit all of the order picking regions respectively. Since these boxes contain products picked for completing an individual customer order, there is no need to sort products.

The advantage of employing pick-to-box order picking system is the reduction in order retrieval time. Complexity of the system and cost problems stem from whether there is sufficient workload assigned to each picking region or not. Moreover, such systems are preferred when numerous and small sized products with low demand quantities are considered for storage.

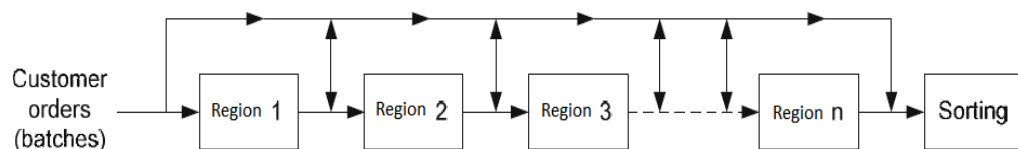


Figure 2.3 Illustration of pick-and-pass order picking systems

2.2.3 Pick-and-Sort Order Picking Systems

In pick-and-sort order picking systems, a batch of customer orders are taken into account. The order pickers traverse the picking region for a particular item. The amount of this particular item to be picked by the order pickers is the sum of the amounts in each individual customer's order. The product group consisting of this particular item is then forwarded to sorting area by the use of conveyor belts. Afterwards, a closed-loop-conveyor is used to sort all of the products and prepare them for delivery to the customer.

Order picking process is started when the number of orders reaches a threshold amount. In "pick-and-sort" order picking systems, the order picking points are visited less frequently than the "picker-to-parts" systems, hence the order retrieval time becomes shorter and the picking efficiency increases. Moreover, it is very important to manage the balance among picking, sorting and packaging activities. Also, implementing an automated sorter requires a considerably high investment. It needs to be noted that pick-and-sort systems become more efficient, when the overlapping and similar orders are considered.

2.2.4 Parts-to-Picker Order Picking Systems

In such systems, an automated vehicle retrieves the ordered products from the storage area and brings them to the order picking area or picking bays. Afterwards, order picker selects required amount of the products. The remaining products are conveyed back to the storage area.

The potential equipment used in the parts-to-picker order picking systems are, carousels, modular horizontal transportation modules, automated storage and retrieval systems (AS/RS).

2.2.5 Completely Automated Order Picking Systems

In such systems, instead of operators, robots and automated machines controlled by computers are used as order pickers. The systems integrated with robotic technology are commonly employed in special cases such as picking valuable, delicate and small items.

Despite the advantages of low rate of product damage, high picking velocity and the ease of integration with the other functions, automated order picking systems require high-level technology, management skills and investment cost (Chang et al., 2007).

2.3 Design and Evaluation of the Order Picking Systems

Companies make decisions on design and control of order picking systems at tactical or operational level, with a different time horizon (Rouwenhorst et al., 2000). According to De Koster et al. (2007), common decisions at these levels are:

- layout design and dimensioning of the storage system (tactical level)
- assigning products to storage locations (storage assignment) (tactical and operational level)
- assigning orders to pick batches and grouping aisles into work zones (batching and zoning) (tactical and operational level)
- order picker routing (routing) (operational level)
- sorting picked units per order and grouping all picks of the orders (order accumulation/sorting) (operational level) (De Koster et al., 2007)

In order to obtain higher performance of the order picking system, modifications and improvements should be made on the factors that affect the performance of the order picking systems; additionally, permanent control of warehouse management system should be maintained.

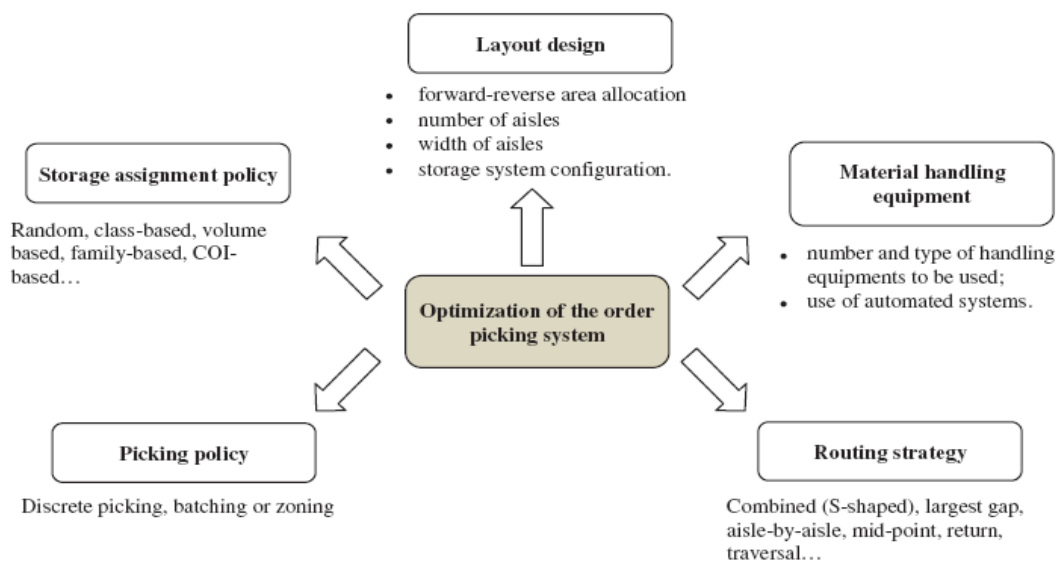


Figure 2.4 Factors affecting the optimization of an order picking system (Bottani et al., 2008)

As it can be observed in the Figure 2.4 above, Bottani et al. (2008), demonstrated the factors used for optimization of the order picking system. In the figure, these factors have been summarized.

Moreover, De Koster et al. summarize the main aspects that affect the order picking system's variety and performance as follows (De Koster et al., 2007):

- Layout Design,
- Inventory assignment/storage policy,
- Routing policy,
- Zoning method,
- Batching method,
- Order accumulation/sorting.

2.3.1 Layout Design

Layout design problem consists of two sub-problems. First problem is known as facility layout problem. This problem includes the determination of how to design

layout together with various departments. In this problem, the most common purpose is to decrease total transportation and handling costs. Accordingly, activity relationships between the departments involving the order picking systems should be considered.

The second problem can also be called as the *internal layout design* or *aisle configuration problem*. It concerns the determination of the number of blocks, and the number, length and width of aisles, the number of cross aisles if they are necessary, the location of the depot (input/output station) as depicted in Figure 2.5. The objective of the problem is to design best layout with respect to given constraints and requirements. As a performance measure, total travel distance and travel time are considered (Dallari et al., 2009).

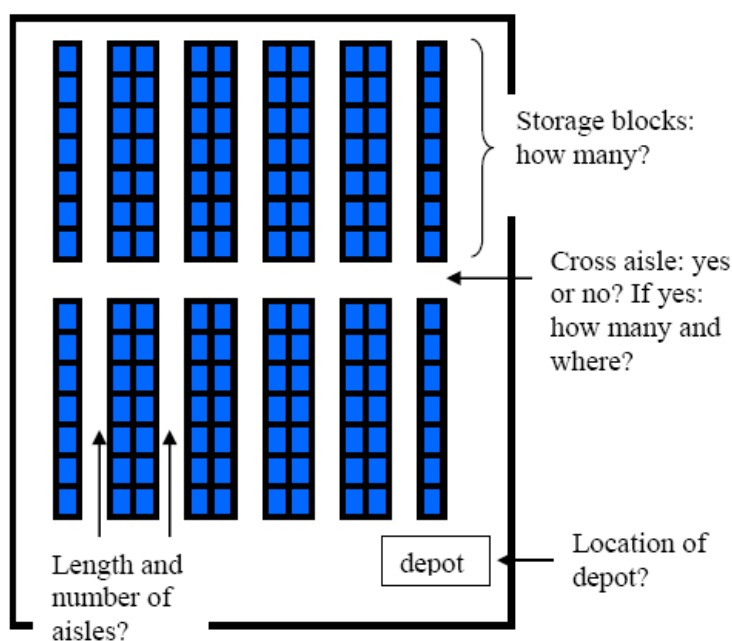


Figure 2.5 General decisions in order layout design (De Koster et al., 2007)

2.3.2 Storage Assignment Policies

There are different ways to assign products to storage areas where they will be kept. Here are the six frequently used types of storage assignment policies;

- Random Storage,
- Nearest empty location storage,
- Dedicated storage,
- Full-turnover storage,
- Class-based storage.

2.3.2.1 Random Storage

A certain amount of palletized similar products are assigned to available empty storage locations randomly with equal probability. Random storage policy incurs high utilization of storage areas, whereas it causes increase in the distance travelled for completing order picking process. On the other hand, this storage policy gives effective results in computer-controlled warehouses.

2.3.2.2 The Nearest Empty Location Storage

Products are stored in the first empty storage location that is encountered by the operators. This generally leads to a warehouse where racks and bays are full near the depot and increasingly emptier towards the back (if there is excess capacity) (Hasuman, 1976).

2.3.2.3 Dedicated Storage

Another storage assignment policy is to store at a pre-determined location, which is called dedicated storage. Every product has an individual storage location dedicated to itself. They can only be stored at these dedicated storage locations. A disadvantage of dedicated storage policy is that a storage location is reserved even for products that are out of stock. So, these storage areas are hold empty until the products are stored. Furthermore, sufficient storage location is allocated for every product, in order to store the maximum level of inventory.

Dedicated storage policy is better to employ for products that have different weights and volumes. For example, light products are assigned to the top shelves of the racks; heavy ones are stored in the low level shelves.

2.3.2.4 Full Turnover Storage

This type of storage approach distributes products over the storage area according to their sales rates which is called as “turnover”. The products with the highest sales rates are positioned at the most convenient and easily accessible locations, usually close to the depot. Products with lower sales rates are located somewhere towards the back of the warehouse.

Heskett et al. (1964) introduced an early storage policy of this type which is called the cube-per-index (COI) rule. The COI value of a particular item is described as the ratio of that particular item’s total required space to the number of tours undertaken to compensate its demand per time interval. The introduced technique consists of positioning items with the lowest COI value closest to the depot. A better and more efficient operation of full-turnover policies would be easiest if integrated with dedicated storage strategy. The main shortcoming of this method is that demand rates fluctuate constantly and the product variety changes frequently. Each alteration would bring a necessity a new ordering of products in the warehouse arising in a large amount of reorganizing of stock. An interpretation might be to execute the restocking once per period. Full-turnover storage policy usage incurs a substantial amount of loss of flexibility and eventually the loss of efficiency. The application of COI-based storage policy, or other policies based on demand frequency generally necessitate a more “information sensitive” method than random storage, since order and storage information must be evaluated in order to classify and allocate products (Caron et al., 1998). In some situations this data may not be accessible, for example, because the product variety changes rapidly to build reliable statistics.

2.3.2.5 Class Based Storage

Class based storage is the storage type in which the products are grouped into multiple classes with respect to either frequency of customer order or relevance of products. Similarly, the storage area is sectioned into various sub-regions, with products in the same group stored randomly in one of the storage regions.

The method of class-based storage combines some of the methods mentioned so far. In inventory control, a classical way for categorizing the items into groups based on popularity is Pareto's method. The idea is to group products into classes in such a way that the highest sales rated class contains only about %15 of the products stored but contributes to about %85 of the turnover. Each class is then assigned to a dedicated area of the warehouse. Storage within an area is random. Classes are determined by some measure of demand frequency of the products, such as COI or pick volume. Highest sales rated items are generally called A-items. The next highest sales rated category of products is called B-items, and so on. Often the number of classes is restricted to three, although in some cases more classes can give additional gains with respect to travel times.

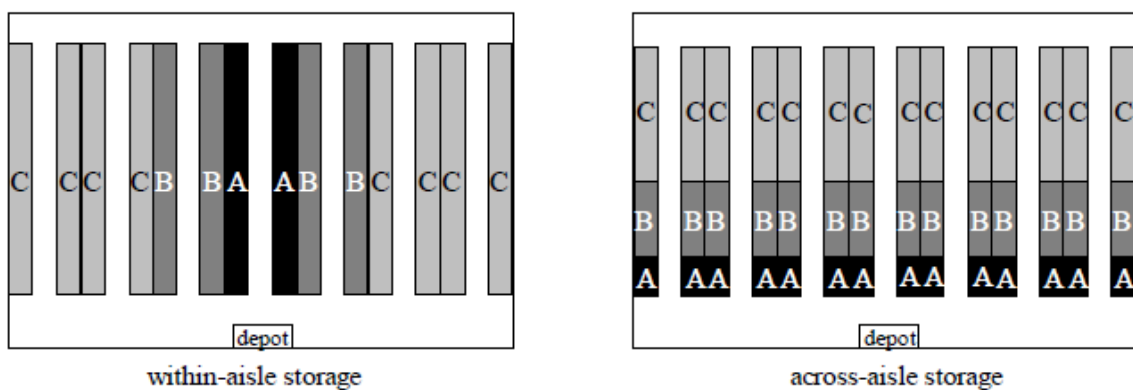


Figure 2.6 Illustration of two common ways to implement class-based storage (De Koster et al., 2007)

2.3.3 Determination of the Order Picking Route

The aim of the order picking route determination is to decide the product picking sequence to generate an efficient path in the forward storage area. The order picking route determination problem is a specific type of the Travelling Salesman Problem.

In Traveling Salesman Problem a list of cities and their pairwise distances are given, and the task is to find the shortest possible route that visits each city exactly once and returns to the origin city. Similarly, the order picker considers the point that it receives the customer order as the origin and traverses the locations in the forward storage area for the items in the order list, finally returning back to the origin. The resemblance of the problem definitions bring out the similarity of Traveling Salesman and Order Picking Problems.

An exemplary warehouse layout and order picking points related to a sample customer order are graphically displayed in Figure 2.7. The black dots on the right-hand-side part of the figure depict the input/output point and order picking locations of a particular customer order. The rest of the dots represent the inter-aisle passage positions which are optional to pass through.

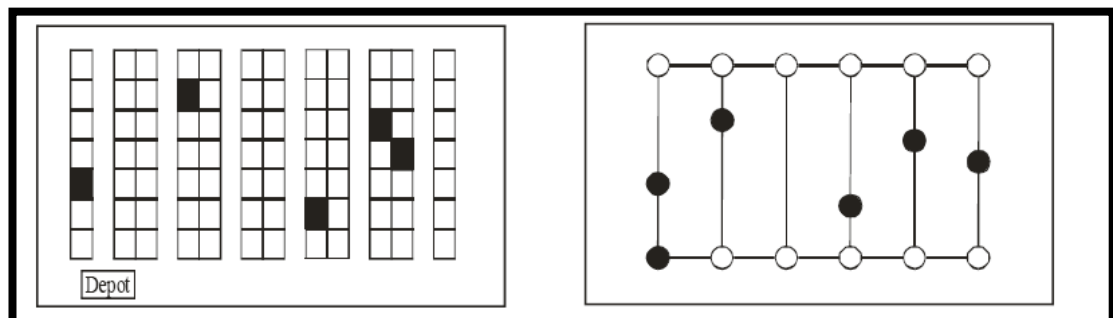


Figure 2.7 Illustration of an order picking system and its graph representation (De Koster et al.,2007)

Order picking route has a great importance on the performance of the order picking system. The sequence of picking the products in the order list, affects various performance measures such as total travelled distance, order retrieval time and the

number of completed customer orders per time unit. Determining the optimal order picking route is a challenging task because of the restrictions like the structure of the warehouse, replenishment policy and traffic congestions in the aisles. Therefore, heuristic methods are often employed in order to determine the picking route. Some of the commonly used methods in the literature are presented in the following subsections.

2.3.3.1 S-shaped/Transversal Heuristic

One of the easiest heuristic methods which is used to determine the order picking route is S-shaped/transversal path selection strategy. The aisle with at least one product in the customer order is traversed entirely along its extent and the necessary items are collected. The aisles which do not have any products in the customer order are not entered or traversed by the order pickers. After picking the last item in the customer order, the order picker exits the aisle and returns to its origin, thus completes its route. In real life practical applications, S-shaped/transversal heuristic method is the most used strategy to determine the order picking route.

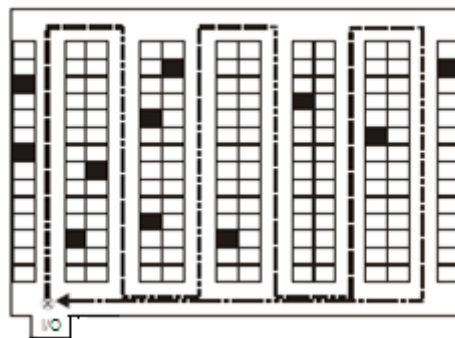


Figure 2.8 Illustration of S-shaped/transversal routing heuristic

2.3.3.2 Return Heuristic

In Return order picking heuristic method, the order picker always uses the same side to enter and exit the aisles while traversing the warehouse for a customer order. Using the Return order picking heuristic method is as easy as utilizing the S-shaped/transversal heuristic strategy. Selection among these two heuristic methods

depends on whether or not to pick from both sides of a particular aisle. If the aisles are designed according to one-side order picking, using Return order picking heuristic method is more appropriate. In this situation, if S-shaped/transversal heuristic method is more appropriate. In this situation, if S-shaped/transversal heuristic strategy was utilized, it would mean to traverse the same aisle twice. If the aisles are designed in a narrow fashion and are suitable for two-side order picking, then using S-shaped/transversal heuristic strategy is more convenient and advantageous. On the other hand, if the aisles are designed in a wider fashion and contain many products from a particular customer order, picking only from one side of the aisle, therefore utilizing Return order picking heuristic method, is more appropriate (Goetschalckx and Ratliff, 1988).

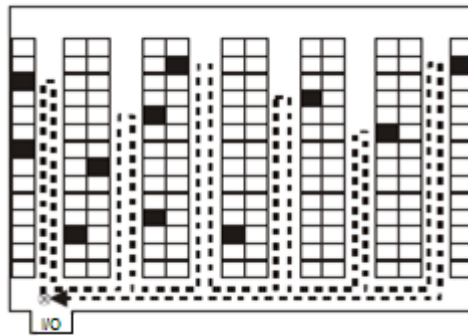


Figure 2.9 Illustration of return routing heuristic

2.3.3.3 Mid-point Heuristic

In Mid-point strategy, the warehouse is divided into two equal sized regions. The products which are in the front region of the warehouse will be picked from the front sides of the shelves, whereas the products which are in the rear region of the warehouse will be picked from the backside of the shelves. The order picker passes to the rear region of the warehouse only at the first or last aisle that is traversed at the front region of the warehouse. In situations where the total order amount per aisle is low (for example one order picking point per aisle), this heuristic method performs better than S-shaped/transversal heuristic strategy (Hall, 1993).

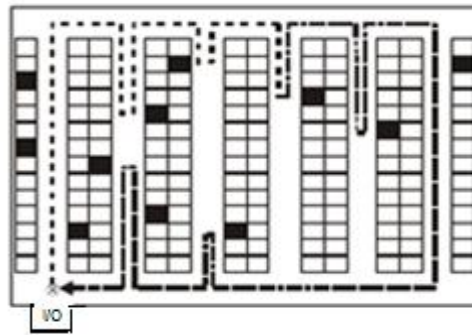


Figure 2.10 Illustration of mid-point routing heuristic

2.3.3.4 Largest Gap Heuristic

In this heuristic method, the order picker enters the aisle, passes through it and exits at the rear side of the warehouse. Inside a particular aisle, the order picker continues to collect the products until it reaches the “Largest-gap” and then exits from the direction it had entered the aisle. The “Largest-gap” inside an aisle can be calculated in three different ways: (i) the distance between two consecutive products inside a customer order (ii) the distance between the entrance of an aisle and the product which is closest to the entrance of that aisle (iii) the distance between the rear side of an aisle and the product which is closest to the rear side of that aisle (Oudijk et al., 1999; Roodbergen and De Koster, 2001a). The highest valued distance that is calculated for an aisle is chosen as the “Largest-gap” inside an aisle. The products which are at the rear sides of different aisles are collected until the order picker reaches the “Largest-gap” inside an aisle. During this process, the order picker first enters and exits the aisle from the rear side of the aisle. When the order picker reaches the last aisle that contains products from the customer order, it passes to the front region of the warehouse. The products which are at the front region of different aisles are collected until the order picker reaches the “Largest-gap” inside an aisle. The order picker, in this time, enters and exits the aisle from the front side of the aisle. In this heuristic strategy, the “Largest-gap” term indicates the distance that is not traversed by the order picker inside the aisles.

2.3.4 Zoning

As an alternative to single order picking, order picking area can be divided into zones. Each order picker is assigned to a zone and performs the picking activity in this zone.

Although zoning method has important effect on the performance of order picking systems, it attracts less attention when compared with the other methods. Advantages of zoning methods are: (i) Order picker travels shorter distance in its own zone. (ii) Occurrence of the traffic congestions are decreased. (iii) Since the order picker knows its own region and the products located, accessibility of the products becomes easier.

Zoning causes splitted orders and this situation requires gathering the products again, after completing the picking operation.

In order to deal with this problem, two approaches are adopted. The first approach is *progressive assembly* of an order. With this approach, order picker only picks the product existing in the order list that is stored in its own zone. After picking, order picker sends the order list to the next zone for completion of the order. In this way, order picking is completed by picking all of the products existing in the order list from the related zone.

Another zoning approach is *parallel* or in other words, *synchronized* order picking method. With this approach, order pickers in each zone, picks the products for the same order list from their related zone. Afterwards, products for each order are gathered and the order picking process has been achieved. Zoning method is practically employed with respect to product properties such as dimension, weight, required temperature and expiry date (De Koster et al., 2007).

2.3.5 *Batching*

When sizes of orders are quite large and each order contains numerous products, discrete picking can be applied and there is no need to batch customer orders. On the other hand, if customer orders are small, in order to reduce the travel time, order batching can be implemented. Using this method, batch of orders can be picked in a single order picking tour (Rouwenhorst et al., 2000).

Order batching is employed according to two different criteria. These are the distances between the products to be picked (*proximity batching*) and the *time windows*. In proximity batching, it is important to calculate the distances between the products existing in the order list. While batching the customer orders, in order to deliver the orders to the customers on time, required attention must be paid to determine the order picking route minimizing the total traveled distance. The aim of the proximity batching is to minimize the order delivery time.

When order batching in time windows is employed, the constant or variable set of orders which came in the same interval or in other words in the same time windows are batched. Order batches are picked in a single tour.

2.3.6 *Order Accumulation/Sorting*

In the use of batching or zoning, additional computations should be executed to divide the batch into several groups. These groups are mainly either the merger of the items per customer order or per shipping destinations of the related customer orders. These series of actions are often called accumulation/sorting (A/S).

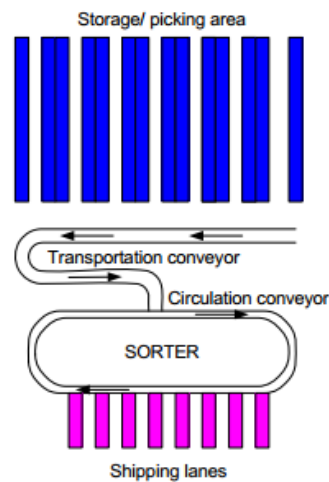


Figure 2.13 Illustration of an A/S system (De Koster et al., 2007)

A common example of an A/S system is shown in Figure 2.13; products of a group of customer orders (a pick-wave) that are to be loaded onto a certain number of trucks are gathered from the forward storage area. Usually, products from the same customer order are dispensed to multiple order pickers (to achieve better order picker efficiency) and the order pickers circulate through fixed routes which are appointed in advance to collect the items assigned to them.

The products that are gathered by the order pickers are placed on the transportation conveyor and are carried to the sorter. Due to the fact that the orders are dispensed to more than one order picker, the products of each order are handled at the sorter in an arbitrary sequence. The sorter, then, releases the products onto the circulation conveyor. Subsequently, the products enter the designated shipping lane if all products of the prior order assigned to that lane have already entered. If not, the products re-circulate around the circulation conveyor. According to the demand of the trucks, the customer orders are released from the shipping lanes. Following that, the capacity of the shipping lanes is adjusted for the consequent sort group. The sorter capacity and conveyor speed are main factors that affect the throughput of an A/S system as well as the operating policies like distribution of orders to shipping lanes (depicted in Figure 2.13). Often, the number of shipping lanes is less than the number of customer orders, which also describes a critical problem for most A/S

systems called order-to-lane distribution problem. This issue may cause a stalling of customer orders at the entrance of the lanes (De Koster et al, 2007).

CHAPTER THREE

LITERATURE REVIEW

In recent years, there is an increase in the number of studies about the design and control of the different type of order picking systems and factors affecting the performance of order picking system.

Solution approaches to the order picking problems can be examined in two main groups, namely, exact and approximate methods. Exact methods usually include the methods that find the best solutions such as branch and bound algorithm and graph based methods like dynamic programming. On the other hand, approximate methods include the heuristic approaches. In the past decade, the researchers have focused on the wide variety of heuristic approaches and these researches were implemented on the automated or manual order picking systems.

Amstrong et al. (1979) handled the order batching problems in semi-automated order picking systems. They developed a mixed-integer programming model with the objective function minimizing the total completion time of an order. Developed model was solved using Benders' (1962) decomposition method.

Ratliff and Roshental (1983) considered a rectangular shaped single-block warehouse in which the aisles are one directional. In other words, if the order picker enters an aisle it cannot return, so it will have to exit that aisle from the other side. They developed a dynamic programming algorithm that aims to minimize the total order retrieval time.

In another study, Goetschalckx and Ratliff (1988) focused on allowing simultaneous movement of more than one order picker in the aisles. Their study identifies the optimal order picking route under the assumption that there is no traffic congestion and that the aisles are inconvenient for two-sided order picking (the order pickers are forced to one-sided picking).

Petersen (1997) tried to develop alternative routing policies for an order picking system employing random storage policy, considering the effects of the factors such as density of the order list, physical structure of the storage area, and location of the input/output point on the order picking systems. In the subsequent study, Petersen and Schmenner (1999) performed the same analysis on several examples of the class-based and volume-based storage policies.

The order picking problems such as inside multi-block warehouses or warehouses that have vertical aisles together with horizontal aisles are addressed by Roodbergen and De Koster (1998).

Caron et al. (1998) concentrated their research efforts on a warehouse with two blocks. In this study, S-shape and return routing heuristics were compared with respect to the expected travelled distance as a performance measure. They further improved their research in their next study (Caron et al., 2000), the researchers determined the most convenient and necessary number of aisles for a particular storage. They considered the same layout, with respect to the density of order list and class-based storage policy.

Vaughan and Petersen (1999) carried out a simulation study determining the number of identical-sized cross-aisles located between the blocks, in order to increase the efficiency of the order picking system. In the order picking system, random storage policy was employed. The authors proposed a dynamic programming based heuristic to determine the order picking route. On the other hand, Roodbergen and De Koster (2001a) extended the same study under the same assumptions, simulating the order picking system for different routing heuristics. For analyzing the performance of the proposed heuristics in the previous study, obtained results were compared with the branch and bound algorithm which is used for determining the shortest order picking route. According to the application results, it is observed that combined heuristic mostly gives better results than the other heuristics. On the other hand, there was a % 1- % 5 variation in optimal results with respect to the size of the order picking list.

Roodbergen and De Koster (2001b) examined the effects of cross-aisle warehouses on the order picking performance. They assumed the location of the cross-aisle in between two blocks of the warehouse and random storage policy was employed. Finally, there is a varying order density inside the warehouse.

In a latter study, the effects of comprehensive combinations of the different order batching policies, storage policies and routing policies were evaluated by Petersen and Aase (2004). Sensitivity analyses were carried out to investigate the effect of order size, location of depot, warehouse shape and demand distribution on the order picking performance.

Roodbergen (2005) developed two different routing heuristics in order to minimize the travelled distance for storage areas involving more than one horizontal cross-aisle. In the study, it was assumed that the storage area has an optimal number of aisles and the items were sent to the production area from only one input/output point.

Le Duc and De Koster (2005) researched the impact of the factors such as the size of the order picking list, storage assignment policies and layout structure. In their study, the assumed warehouse involved only one central horizontal cross-aisle and employed class-based storage policy. For each different order picking system which combines different storage assignment policy, warehouse layout and size of the order picking list, a mathematical model was proposed and a heuristic method was presented to solve the problem. A computational study was conducted to test the performance of heuristics.

In the study based on artificial intelligence techniques, Manzini et al. (2005) represented an expert system integrated approach used for evaluation of the order picking system. This approach combines simulation, genetic programming and statistical analysis methods. It examines the effects of order content, layout of the warehouse, routing and storage policies, on the order picking performance both for

“picker-to-parts” and “parts-to-picker” systems. Researchers defined the performance measure as the average cycle time of completion of an order.

Hwang and Cho (2006) proposed a two-stage solution method for the design problem of order picking systems. First, a mathematical model minimizing the total cost of order picking was developed. While developing this model, issues such as compulsory demand constraints and storage capacity of the warehouse were taken into account. Secondly, while designing the layout of the storage area, effects of the factors such as the size of the warehouse, dimension of the racks, number of order pickers in the system were examined on the performance of the order picking system with the simulation study.

Roodbergen and Vis (2006) tried to develop the most convenient layout design for the storage area that has a single block. While applying the random storage policy, S-shaped and largest gap routing heuristics were compared. In the study, a non-linear mathematical model with the objective function that aims to minimize the average order retrieval time was proposed. Observing the application results, it was proved that there is a strong relationship between the optimal number of aisles, the size of order batch, order picking route and dimension of the required storage area.

Parikh and Meller (2008) developed a cost model for selecting the suitable method between the order batching and zoning methods. With this model, factors such as imbalance in the order load to be picked, order picking velocity, traffic congestions that may occur in the aisles and the necessity for sorting the picked items were evaluated.

In the literature, it is usual to encounter the order picking systems containing only one order picker. Aside from these systems, Pan and Shih (2008) dealt with the problem of traffic congestion in the storage area containing more than one order picker. In this study a queueing network is proposed in order to determine picking performance and consider a trade-off between travelled distance and blocking-caused delay. The proposed model is validated by a simulation experiment and used to

compare the efficiency of order picking system under different storage assignment strategies. Researchers tried to increase the number of completed customer orders per time unit (throughput) while preventing the traffic congestions that may occur in the aisles.

Pan and Wu (2009) developed an analytical model for pick-to-box order picking system which is usually applied when e-commerce or time-based competition are considered. In order to estimate the total travelled distance, operations carried out by the order pickers were defined as Markov Chain. Based on the developed model, three different algorithms were employed to assign the products to the most convenient storage area and their results were evaluated. In the study, effects of frequency of order picking, number of traffic congestions in the aisles and sorting of the picked products were analyzed simultaneously.

In another study, Yu and De Koster (2009) developed a queuing network model to analyze the effects of order batching and zoning strategies on the order retrieval time. Proposed order picking system was classified as pick-to-box order picking system and the S-shaped/transversal and random storage policies were implemented in the system. Developed queuing model includes the sorting of the products according to the individual customer order. According to application results, proposed queuing network model gives acceptable results until a certain number of order picking zones and utilization rate are reached. When the number of order picking zones increases, variations in the results also increase accordingly.

Bindi et al. (2009) considered the storage policy based on the method of grouping similar products. This policy was evaluated according to the similarity index and family grouping techniques used for grouping the products. And the performances of these techniques were compared in the study.

Roodbergen and Vis (2009) surveyed the existing models and solution methods in the literature. They investigated about the issues such as design of automated storage

and retrieval systems, storage assignment and sequence of the order requests in “parts-to-picker” order picking systems.

In another study that examines the “parts-to-picker” order picking systems, Lerher et al. (2010) developed different analytical models considering the acceleration and deceleration parameters of the automated storage and retrieval systems. With a simulation study, performance of the proposed models was evaluated with respect to the total order retrieval time.

Hsieh and Huang (2011) developed two different batch construction heuristics: K-means Batching and Self Organization Map Batching. The authors compared the proposed batching methods in terms of the average utilization of the transporter used for order picking and total travelled distance. In order to obtain the most suitable combination of the routing heuristics, storage policy and order batching method, a simulation study under different system configurations was implemented.

In their latter study, Lerher et al. (2011) built up analytical models on the design of the “parts-to-picker” order picking systems. The significance of the research was that they studied on the warehouses that the products were carried by the containers (mini-load storage systems) and multiple load-carrying vehicles were used. The proposed system is used for storing small-volume items. The conveyor belt run was integrated with the automated stowing appliances. The performance of analytical models that were introduced in the study was compared by the simulations which were performed for various storage layouts.

Chan and Chan (2011) discussed a real-life problem. In this problem, a warehouse with multi level rack system was considered. In the order picking system, picking was carried out by the operators. A simulation model representing the system was developed. Afterwards, using the model, for alternative storage policies (random storage policy, class-based storage policy and dedicated storage policy), routing heuristics (S-shape heuristic, return heuristic and combined heuristic), different order

density levels, and different system configurations were compared with respect to the total travelled distance and order retrieval time.

In a recent study, Pan et al. (2011) considered “picker-to-parts” order picking systems which employ more than one order pickers. They evaluated the effects of different storage policies on the performance of order picking systems concerning the total travelled distance and order retrieval time. The order picking system was modeled as a queuing network. In order to reduce the waiting time and total travelled distance emerging from the blocking in the aisles, the workloads for picking in the aisles have been tried to balance with the proposed heuristic algorithm. Also, using the simulation model developed, the efficiencies of different storage policies were compared.

Comprehensive literature reviews about design and control of order picking systems in warehouse management were carried out by De Koster et al. (2007), Gu et al. (2007) and Dallari et al. (2009).

When the studies in the literature have been reviewed, it is realized that there is a lack of research about the effect of aisle configuration on the order picking efficiency. Similarly, number of studies about layout design of low level, manual order picking systems is limited in the literature. Moreover, in most of the studies order picking systems were considered as static systems; inventory replenishment was not taken into consideration (Tuna G. and Tunçel G., 2012). Hence, it was assumed that there was no shortage occurred in the warehouse during the operation process, which is not a realistic case in practice.

CHAPTER FOUR

EVALUATION OF AN ORDER PICKING SYSTEM USING SIMULATION

4.1 Problem Statement

In this study, a hypothetical distribution warehouse has been studied in order to improve the order picking performance by developing alternative combinations of routing and storage policies. This warehouse has been designed considering the real-life warehouse of a company specialized in the production of fasteners.

Order picking is very important for the success of the company due to the fact that the shorter the lead time, the company gets the opportunity to compete in the market. Currently, there is no organized order picking system in the warehouse. Products are assigned to the storage locations randomly, but there is not a proper routing policy applied. These conditions make it difficult to achieve the opportunity of sales hit and meet the customer demand on time. So, current order picking system needs to be modified and evaluated.

The warehouse is a manual pick, multi level warehouse and order picking system is classified as low level picker-to-parts order picking system since the order picker drives along the aisle to pick the products in the order list and it is low level system as the order picker retrieves the requested items from storage racks or shelves, while travelling along the storage aisles. In addition, picking process is performed manually by an operator driving a forklift.

There are 20 different types of products stored in the warehouse and every product has a date of production. The products with older production dates have priority in the picking process since the products tend to be oxidized.

The warehouse consists of 4 aisles and 8 racks. Each rack has 5 levels. There are 4 bays in each level. Figure 4.1 presents the vertical plan for a part of the warehouse involving four aisles and 8 racks.

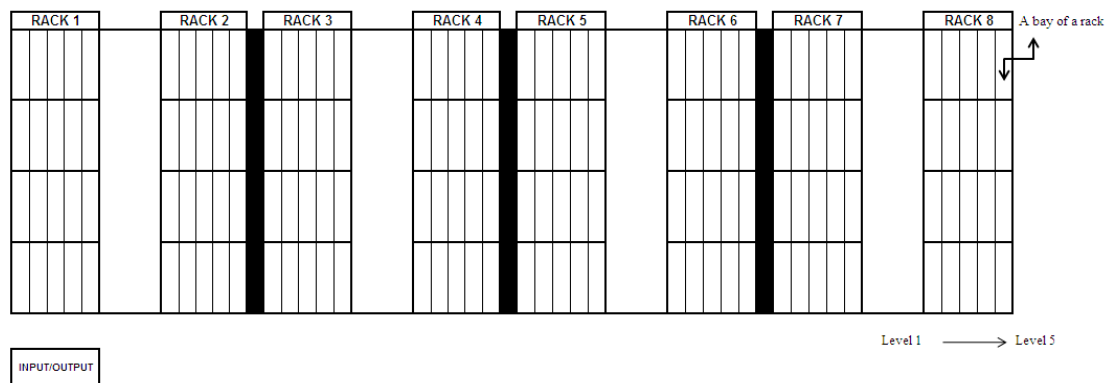


Figure 4.1 Configuration of the warehouse

4.2 Solution Methodology

In the design of new production facilities, warehouses and distribution centers, a successfully used tool is Simulation. It is also used to evaluate suggested improvements to existing systems.

In this study, simulation is used for exploring the picking efficiency under different combinations of storage assignment and routing policies and for comparing the performances of these policies. Statistical analysis is used to analyze and present the findings relevant to the objectives of the simulation study.

The objective of the simulation study is to determine the combination of factors providing the shortest order retrieval time. Order retrieval time is the time elapsed for the process of retrieving products from storage area to meet a specific customer demand. Two models of the same warehouse layout are built considering different routing policies (i.e. S-shaped/transversal, return and mid-point) and different storage policies (random and dedicated storage). ARENA 10.0 has been used to build the model for analysis.

4.2.1 Implementation of the Simulation Model

The following assumptions are made for the simulation study:

1. All products are stored in the same sized boxes.
2. Each bay contains only one product type.
3. No shortage occurs in the warehouse since the system is dynamic and inventory replenishment is performed according to (s, S) policy. The operating policy is employed by defining two numbers, s and S , to be used as follows: When the level of on hand inventory (the total quantity of the product in the warehouse) is less than or equal to s , an order for the difference between the on-hand inventory and S is placed. If u is the total quantity of product existing in the warehouse at any time, then the (s, S) policy is
 If $u \leq s$, order $S-u$.
 If $u > s$, do not order.
 In the simulation model (s, S) values are parametric and change according to the product type stored in the warehouse. In this problem, (S, s) values have been taken as (300, 800).
4. When dedicated storage is employed, products are grouped aisle by aisle according to their similarities. Similar product types are stored in the same aisle and every individual product type is stored at the same level of the opposite racks.
5. Different product types are represented by different numbers in the simulation model (i.e. 1, 2, 3, 4.....20).
6. The input/output point (depot), possible load/unload and entrance/exit points of the each aisle are graphically depicted in Figure 4.2. As it can be seen in the figure, the green points represent the entrance of corresponding aisle, the red ones represent the exit of corresponding aisle and the other points except the input/output point refer to the load/unload points. For example, A11 corresponds to the first load/unload point of the aisle 1, whereas E3 in red represents the third exit point from aisle 3. These points are used as decision points while routing the transporter through the aisles.

7. An operator driving a forklift is employed as an order picker. Velocity of the forklift is 10km/h.
8. Loading-unloading times are uniformly distributed with a range of 3 minutes to 5 minutes.

According to these predefined assumptions and parameters of the order picking system, simulation models were developed for the proposed analysis.

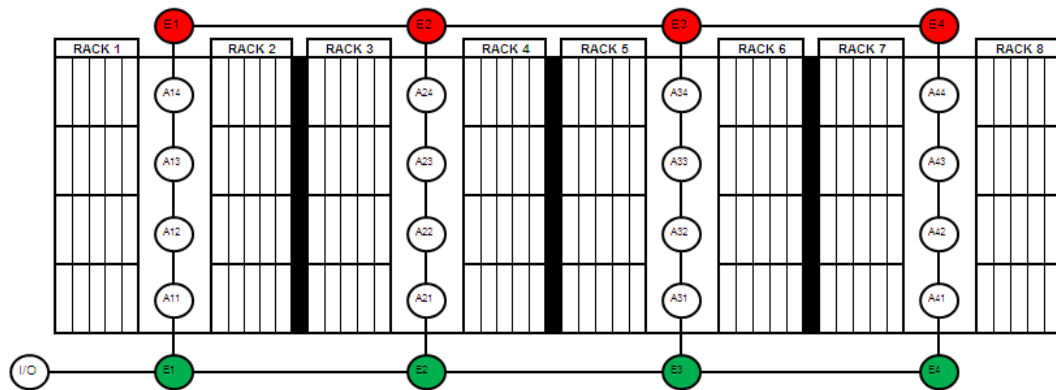


Figure 4.2 The graphical representation of the decision points.

4.2.1.1 Algorithm of the Simulation Model

In the flowchart depicted in Figure 4.3, general algorithm of the simulation model is described. The flow of the algorithm is conducted as follows:

When a simulation run is executed, an order arrival is created as an entity and joins the queue for order picking lists waiting for completion. At $t=0$ it is assumed that all of the storage areas are full and type and production date of the each product in the warehouse are known. Since the stochastic nature of the system, orders are created randomly.

Determining the content of the order list is conducted by using of Submodel 1. The flow chart of Submodel 1 is depicted in Figure 4.4. First of all, size of the order list is derived from a uniform distribution with a range of 1 to number of product types exists in the warehouse. Then, for each product type a random value is derived

from a uniform distribution with values 1 through 10, and these values are assigned to the each individual product type. Afterwards, all of the product types join a dummy queue and are ranked according to the LVF ranking criteria with respect to the assigned random value. From the dummy queue, number of different product types with respect to the size of the order list is selected and added into the order list. For every product in the order list, a demand size is derived from a uniform distribution with a range of 1 to 100. So, an order list which contains the information of the ordered product types and demand sizes is created randomly.

Storage assignment policy and routing policy of the order picking system are determined.

Subsequently, for each type of products, (s, S) policy is checked. If it is necessary, replenishment procedure is executed for the required type and required amount of the product in the list. Submodel 2 is run for this process. Its flow chart is depicted in Figure 4.5 in detail. If replenishment is necessary, it is calculated the required amount of product by subtracting the total amount of corresponding product in the warehouse at t_{now} from the respective S value. At time $t_{now} + t$ for storing the new products, empty bays are searched. If the storage policy is chosen as the ‘dedicated storage’, new items are stored in the dedicated area for the proposed type of the item. Otherwise, if the storage policy is random, products are stored in the available empty bays randomly.

Afterwards, order picking route is determined as follows: Each bay that is used for the storage in the warehouse is represented by an index in the simulation model. The warehouse is represented as a matrix in which the levels of the racks correspond to the rows of the matrix and bays correspond to the elements of the matrix. When a pick is to be made; starting from the first level of the first rack or the first row of the matrix, the bay that contains the required product with the oldest date of production is selected. If sufficient amount of the product to meet all the demand from that particular bay does not exist, it can be met from another bay that satisfies both the type and the production date requirements. According to this procedure, each bay

index, that is selected to complete the customer demand, is first encoded and later added into route list as a single value. Later, a forklift as a transporter is requested to complete the individual order picking list. In accordance with the storage and routing policies, order picking process is accomplished and the forklift returns to the depot to handle a new order picking list and so on. Finally the time elapsed between t_{now} and the time at a transporter was requested is calculated and recorded as order retrieval time for an individual order.

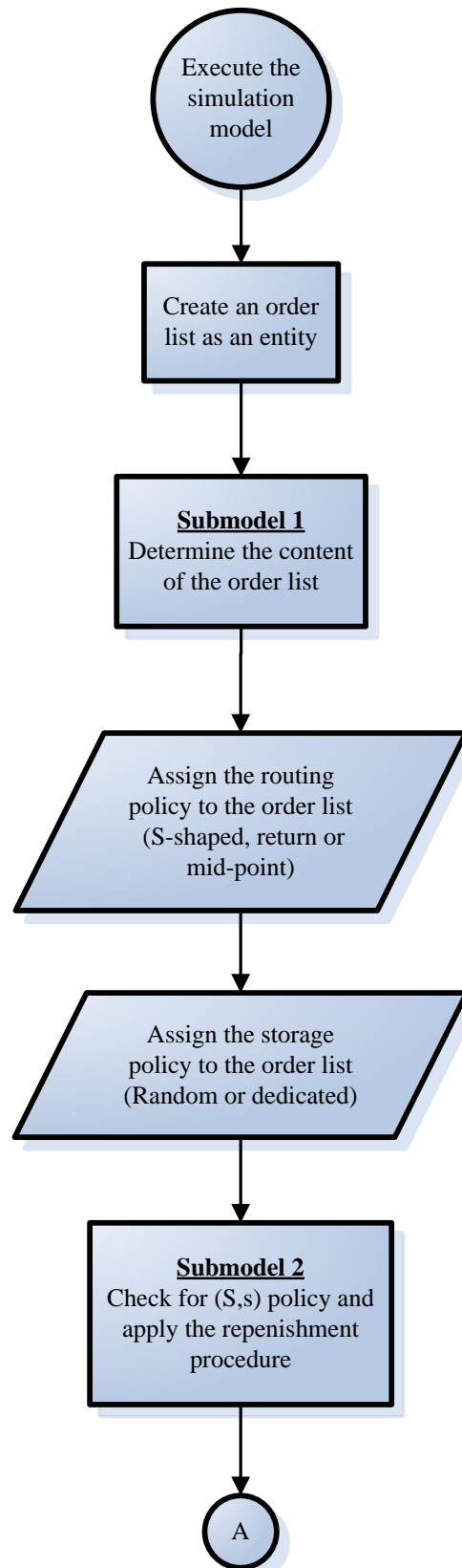


Figure 4.3 The general algorithm of the simulation model

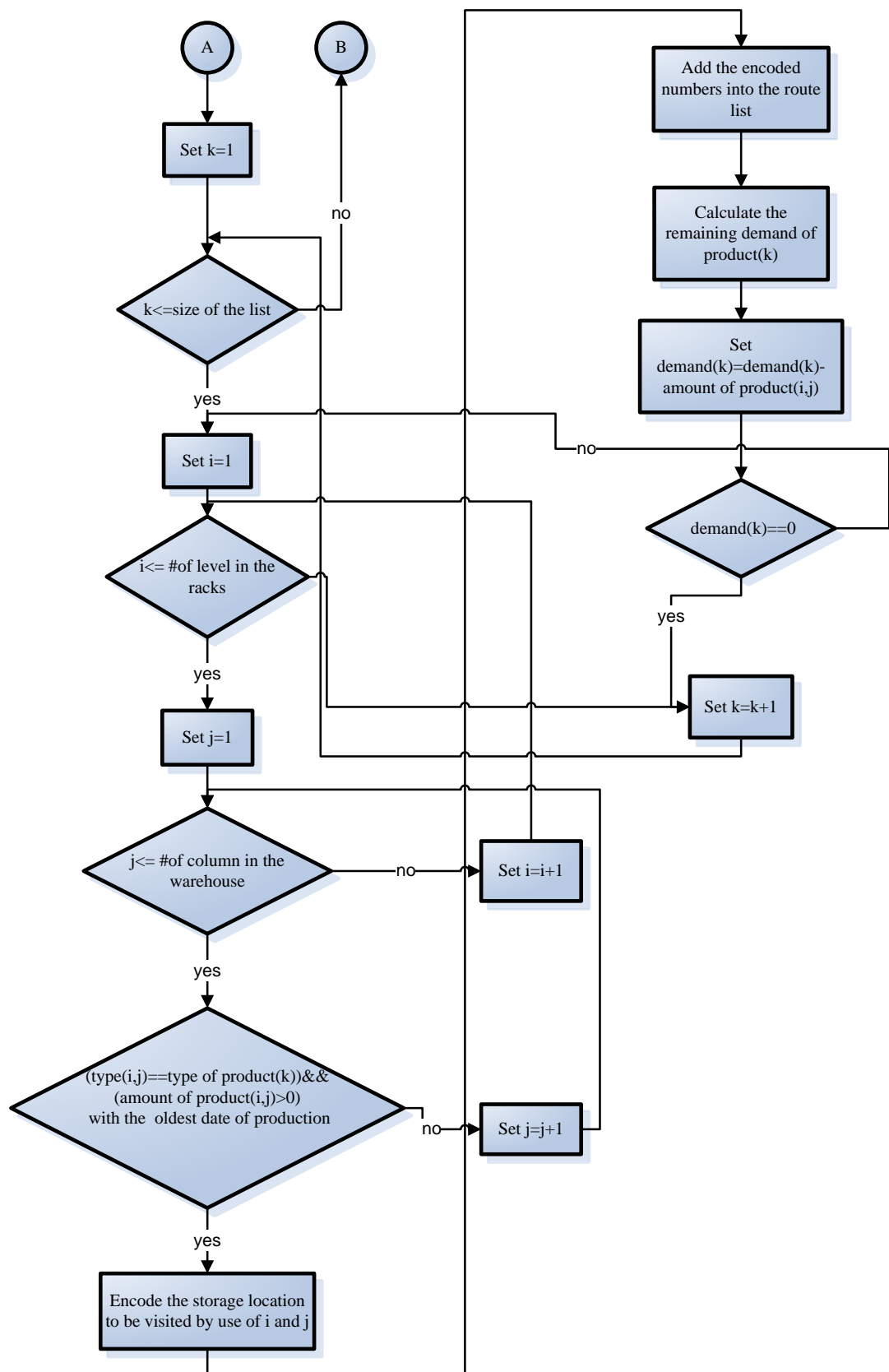


Figure 4.3 The general algorithm of the simulation model (continued)

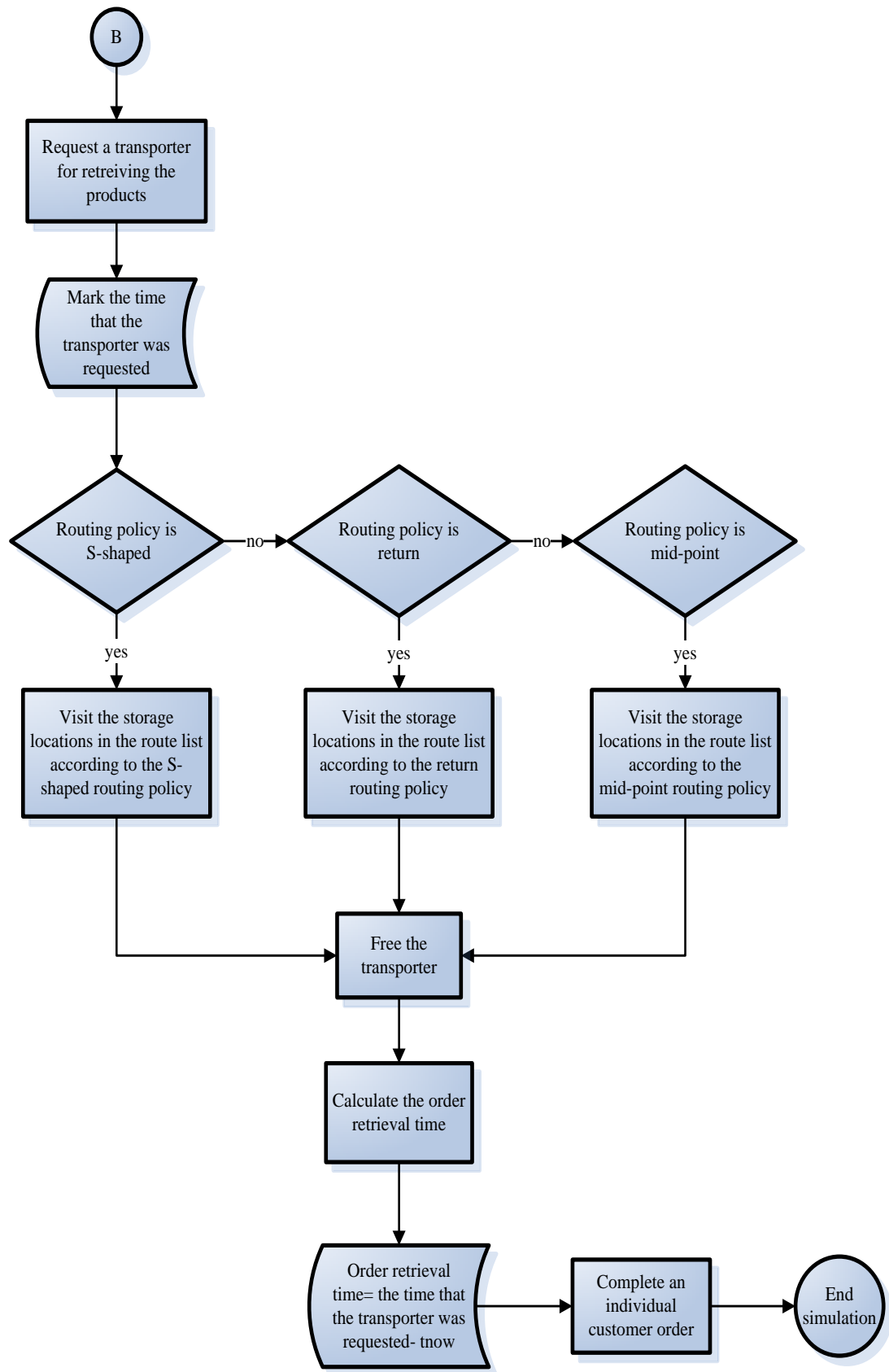


Figure 4.3 The general algorithm of the simulation model (continued)

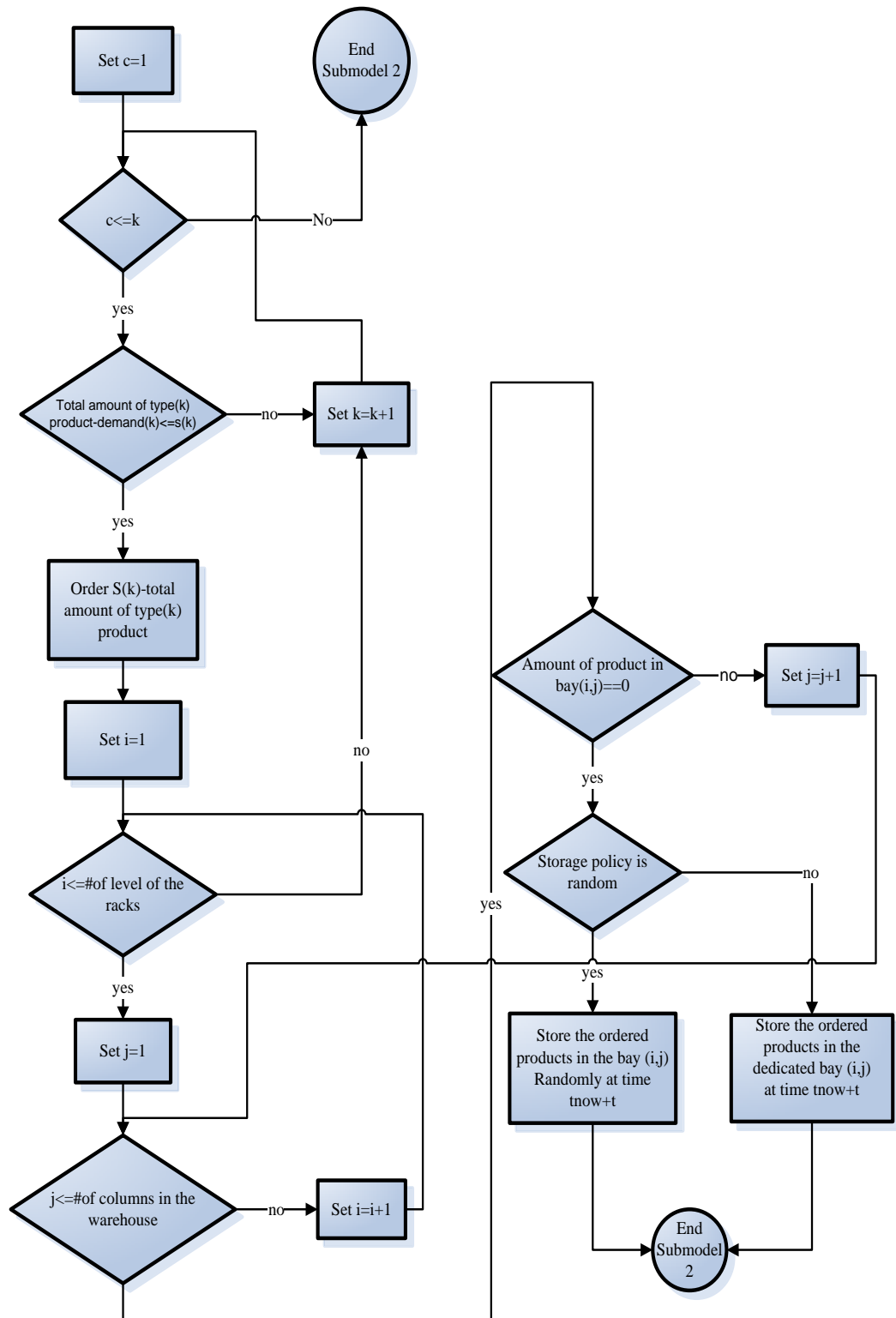


Figure 4.4 Algorithm of the submodel 1

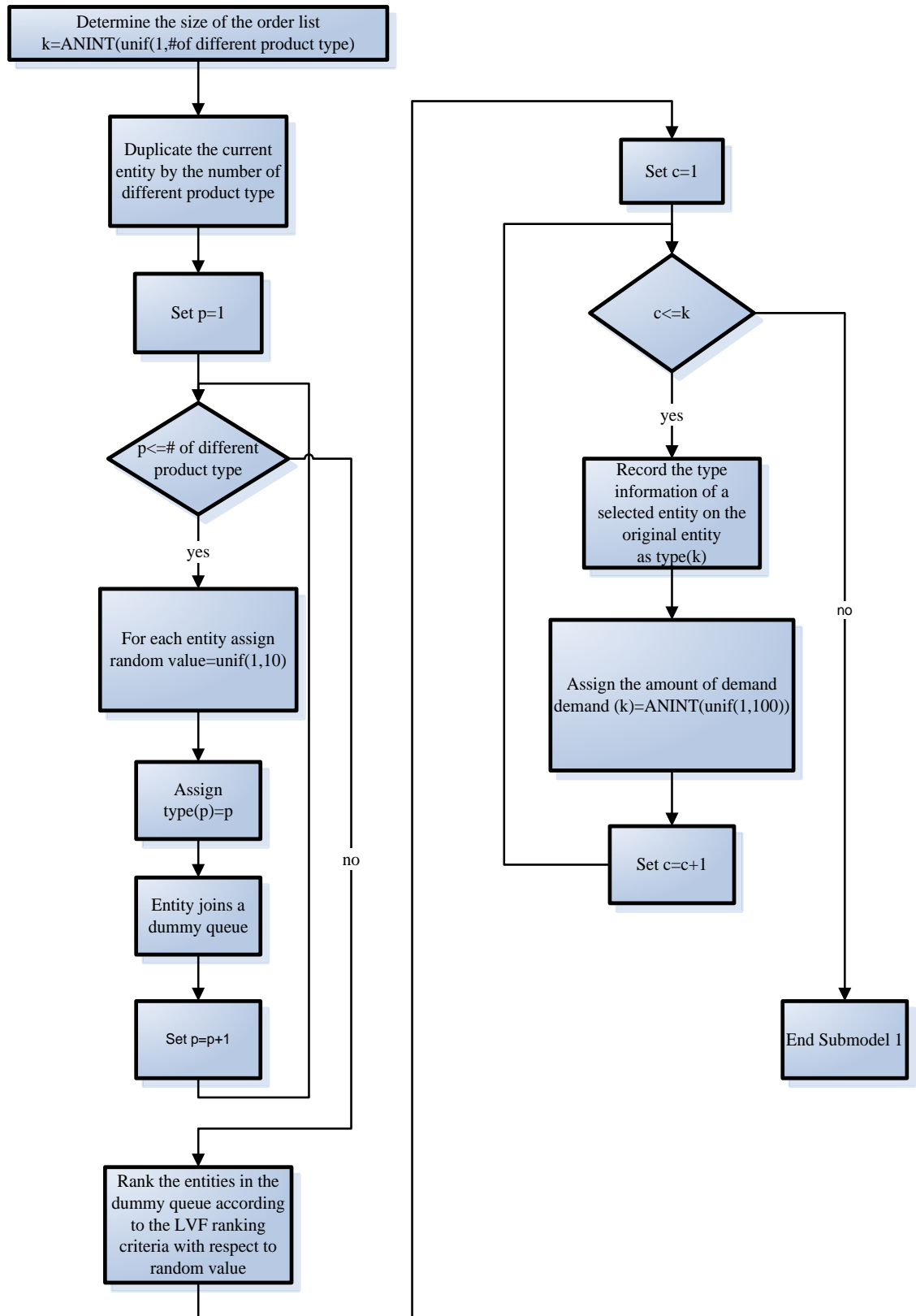


Figure 4.5 Algorithm of the Submodel 2

4.3 Output Analysis

The simulation model has been executed for six different order picking systems. Assumptions, methodology and implementation presented in the previous sections have been taken into consideration. Different systems have been composed to apply the different storage policies with alternative routing policies together. These systems are represented in the following figures. Different letters indicate different product types.

- System 1 is the combination of the random storage policy and the S-shaped routing policy and depicted in Figure 4.6.

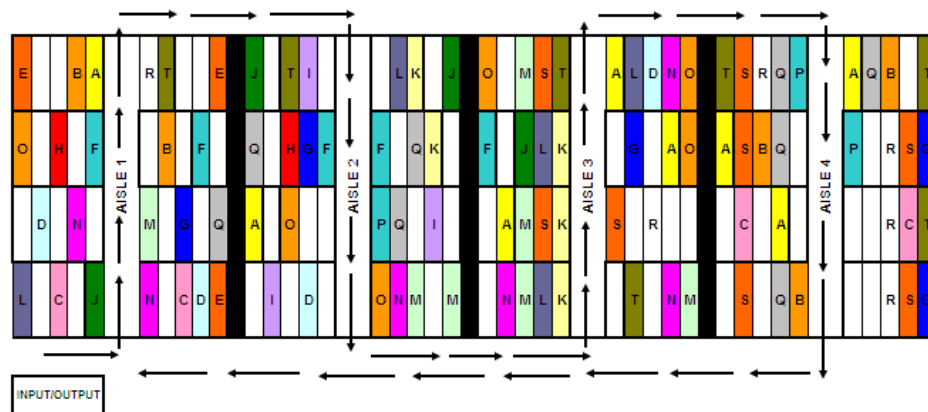


Figure 4.6 Configuration of order picking system 1

- System 2 is the combination of the random storage policy and the return routing policy and depicted in Figure 4.7.

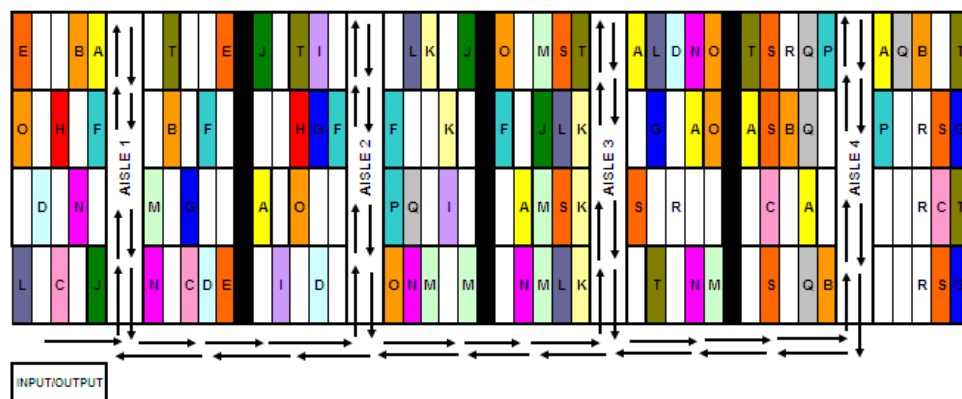


Figure 4.7 Configuration of order picking system 2

- System 3 is the combination of the random storage policy and the mid-point routing policy and depicted in Figure 4.8.

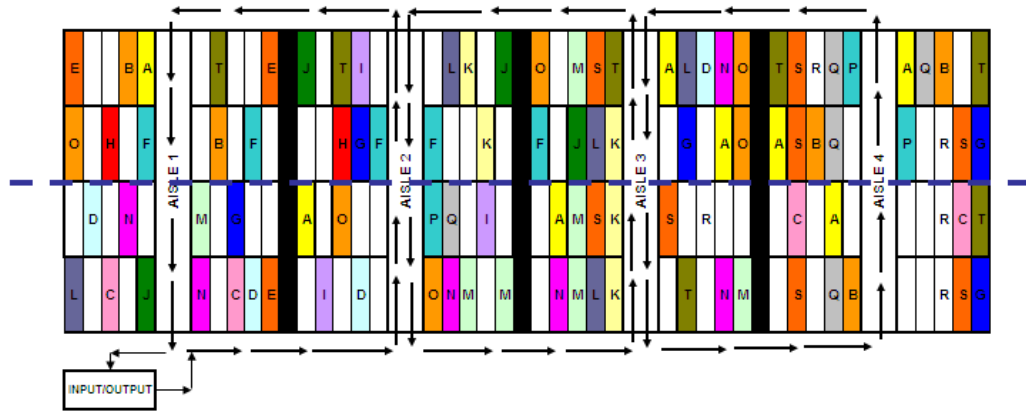


Figure 4.8 Configuration of order picking system 3

- System 4 is the combination of the dedicated storage policy and the S-shaped routing policy and depicted in Figure 4.9.

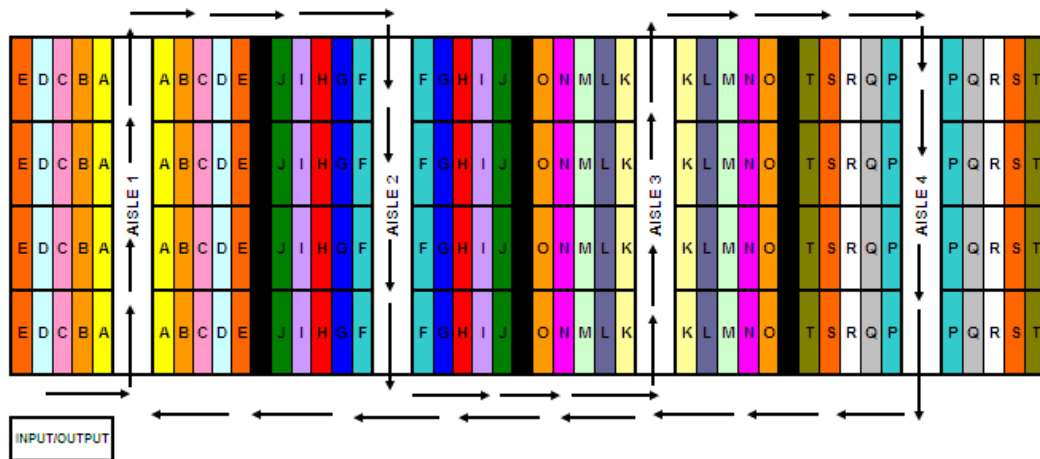


Figure 4.9 Configuration of order picking system 4

- System 5 is the combination of the dedicated storage policy and the return routing policy and depicted in Figure 4.10.

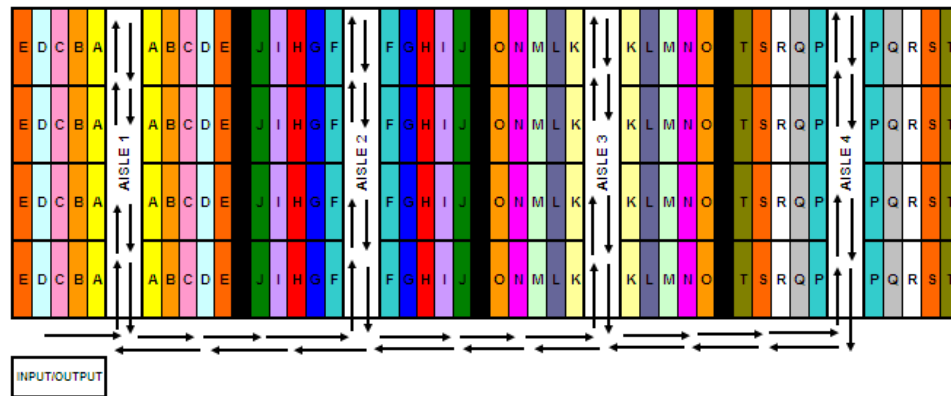


Figure 4.10 Configuration of order picking system 5

- System 6 is the combination of the dedicated storage policy and the mid-point routing policy. System 6 is depicted in Figure 4.11.

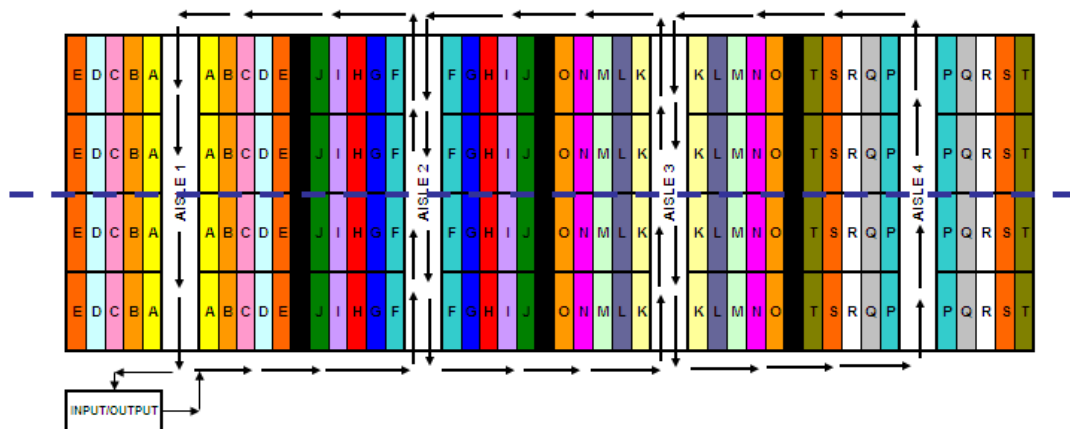


Figure 4.11 Configuration of order picking system 6

4.3.1 Verification and Validation of Simulation Model

Model building is expected to have errors. Model builder is inclined to make mistakes and face error messages during the execution of the model. Verification and validation are used to reduce and ideally eliminate these errors.

Verification is concerned with building the model correctly. It proceeds by the comparison of the conceptual model to the computer representation that implements that conception. It also asks some questions: Is the model implementation in the

simulation software correct? Are the input parameters and logical structure of the model representations accurate? These questions can be answered in various ways. A verified model is a “bug free” model and it doesn’t give any error messages such as syntax errors (unintentional notation errors) or semantic errors (logical errors). Also, it can be determined whether the simulation model runs correctly by reviewing the model code, checking for reasonable output values, observing the animation of the model and using the trace and debugging facilities.

Validation is concerned with building the correct model. It attempts to confirm that a model is an accurate representation of the real system. It is employed by observing the animation, comparing the model with the actual system and the other models and testing against the historical data.

In this study, the validation and verification of the model have been completed before using it to make implementation decisions.

Verification is employed by reviewing the simulation model’s blocks and elements, using the trace and debugging facilities. The model has been validated by observing the animation, using trace element and checking for reasonable output. After verification and validation of the simulation model, it has been decided that the model runs correctly as intended. Moreover, the model is said to be an accurate representation of the real-life warehouse and the order picking system considered in this study.

4.3.2 Determining the Length of Warm-up Period

Terminating simulations and steady-state (non-terminating) simulations can be diversified while analyzing the simulation output data. If we are interested in the behavior of the system over a particular period, we use terminating simulation. Otherwise, if we are interested in the steady-state behavior of a system, we use non-terminating simulation.

In terminating simulation case, simulation starts at a defined state or time and ends when it reaches some other defined state or time. Since the simulation model in this study starts with an empty system and terminates when a certain number of customer orders are completed and arrival process is terminated at a pre-determined time, simulation model is classified as a terminating simulation.

The initial state for the simulation model must be determined since a particular initialization of the model creates bias in the output of the simulation model. Another reason is that the comparison of more than two systems will be carried out in the following parts of the study. Initial state for the simulation model has been selected by determining the warm-up period.

Welch Procedure has been applied to decide warm-up period and observe the steady state condition. The simulation model has run for one replication and 500 order picking lists. Time in system was plotted in Figure 4.12 Time in system represents the time elapsed between the arrival and completion of a particular order.

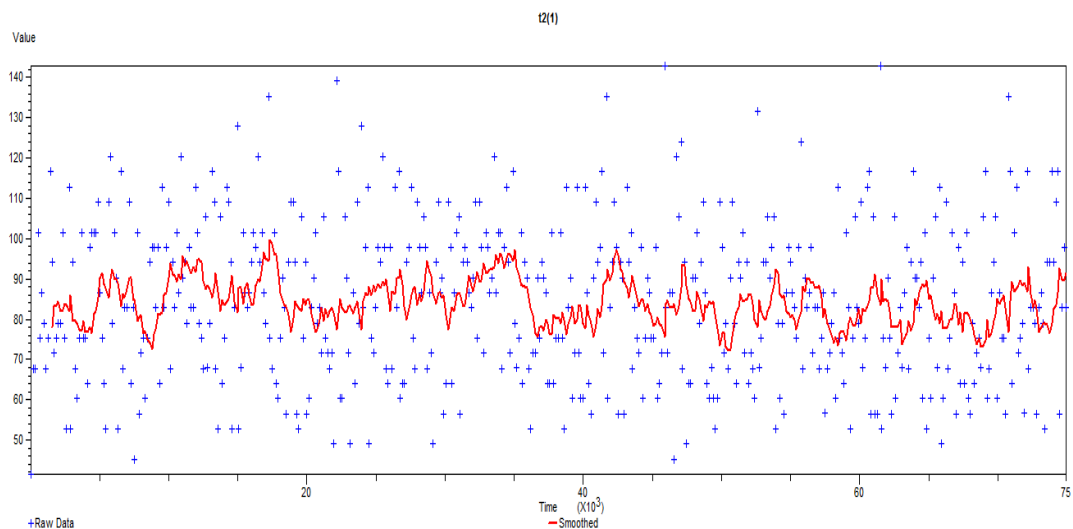


Figure 4.12 Time in system for order picking list

It is clear from Figure 4.12 that further smoothing of the plot is necessary. Hence, further smoothing was achieved using moving average method with time window $w=10$ and $w=100$ and plotted in Figure 4.13 and Figure 4.14, respectively. From the

plot with $w=100$, the smoothest of the plots, the warm up period has been determined as 15000 minutes.

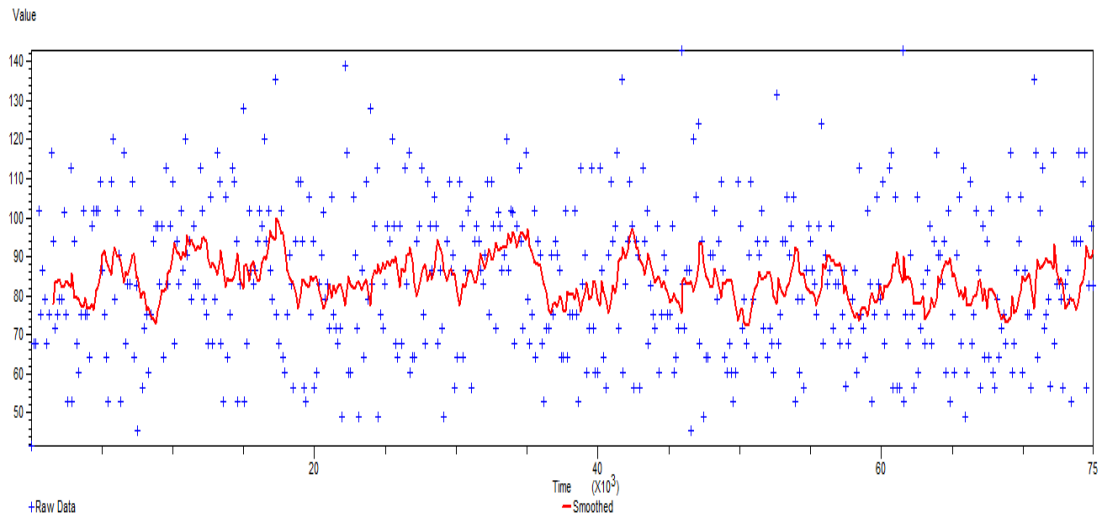


Figure 4.13 Moving average with $w=10$ for time in system for order picking list

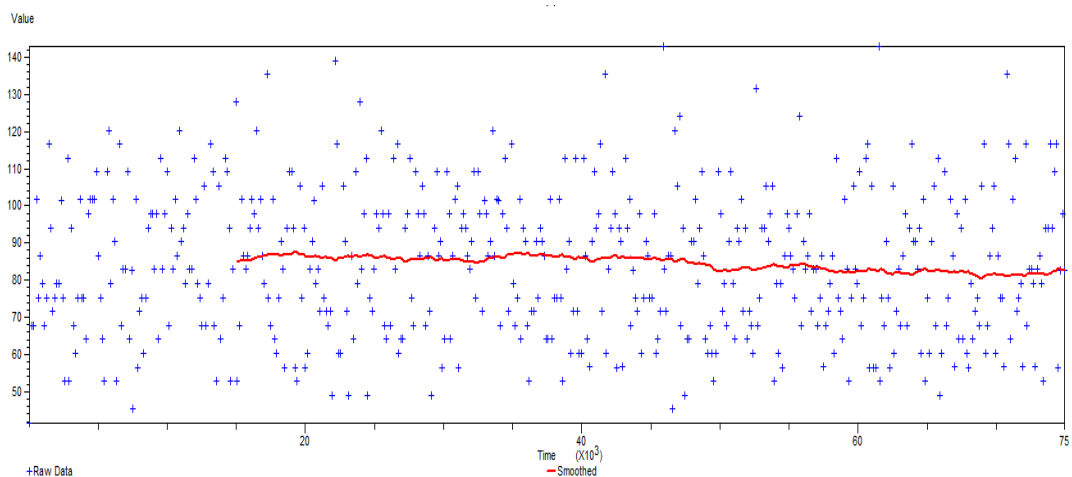


Figure 4.14 Moving average with $w=100$ for time in system for order picking list

Finally, meeting all demands for 500 order picking lists is selected as a terminating event. The number of replications is chosen as 10.

4.3.3 Confidence Interval Estimation

A confidence interval is a range within which we can have a certain level of confidence that the true mean falls. Confidence interval estimation provides information about how far we deviate from the point estimate.

Suppose that we would like to obtain a point estimate and confidence interval for the mean $\mu = E(X)$, where X is a random variable defined on a replication as average order retrieval time. Make n independent replications of the simulation model and let X_1, X_2, \dots, X_n be the resulting independent and identically distributed (IID) random variables. Then by using the equation 4.1 we get that $\bar{X}(n)$ is an unbiased point estimator for μ and standard deviation is calculated by using of equation 4.2, so an approximate $100(1 - \alpha)$ percent ($0 < \alpha < 1$) confidence interval for μ is given by equation 4.3. After the determination of the interval, it can be said that we are $100(1 - \alpha)$ confident that the true value of the parameter is in our confidence interval.

$$\bar{X}(n) = \frac{\sum_{i=1}^n X_i}{n} \quad (\text{eq. 4.1})$$

$$S^2(n) = \frac{\sum_{i=1}^n (\bar{X} - X_i)^2}{n-1} \quad (\text{eq. 4.2})$$

$$\bar{X}(n) \pm t_{n-1, 1-\frac{\alpha}{2}} \sqrt{\frac{S^2(n)}{n}} \quad (\text{eq. 4.3})$$

Using Arena Output Analyzer, according to the methodology mentioned above and considering the outputs of the simulation models, it has been individually estimated 95% confidence interval for mean order retrieval time for all proposed

order picking system. From Figure 4.15 to Figure 4.20, all confidence intervals are depicted for systems 1 through 6. In the figures, parameters such as average, standard deviation, half width, minimum and maximum values for mean order retrieval time can be observed.

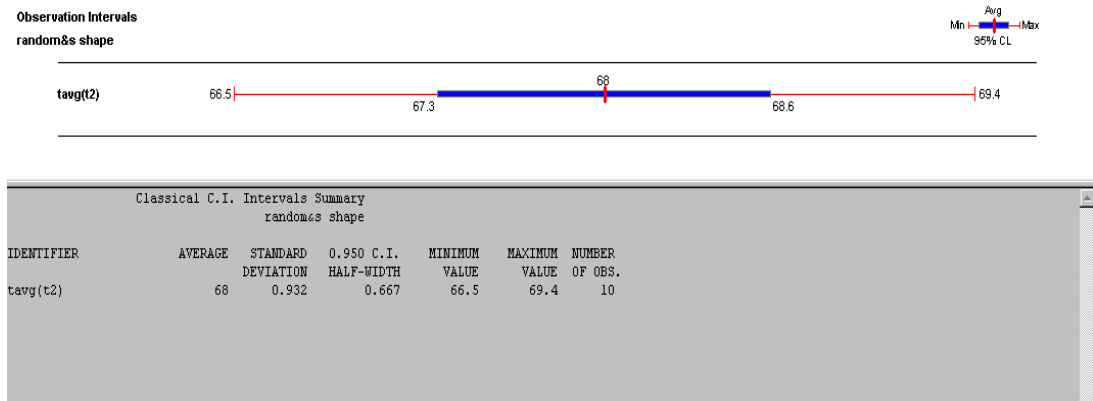


Figure 4.15 95% Confidence interval for mean order retrieval time for system 1.

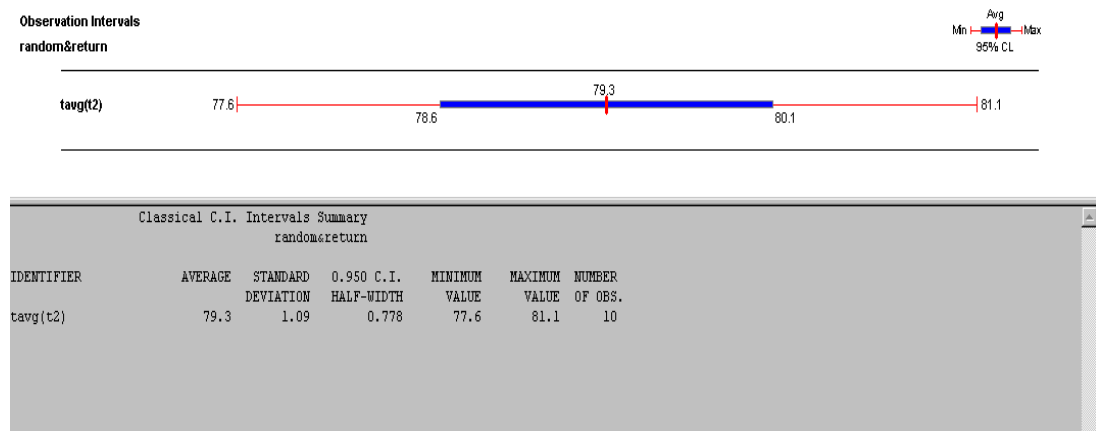


Figure 4.16 95% Confidence interval for mean order retrieval time for system 2

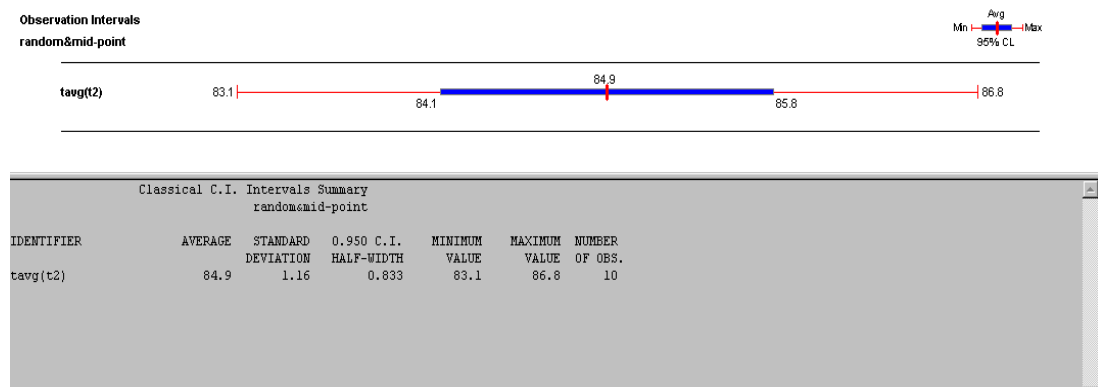


Figure 4.17 95% Confidence interval for mean order retrieval time for system 3

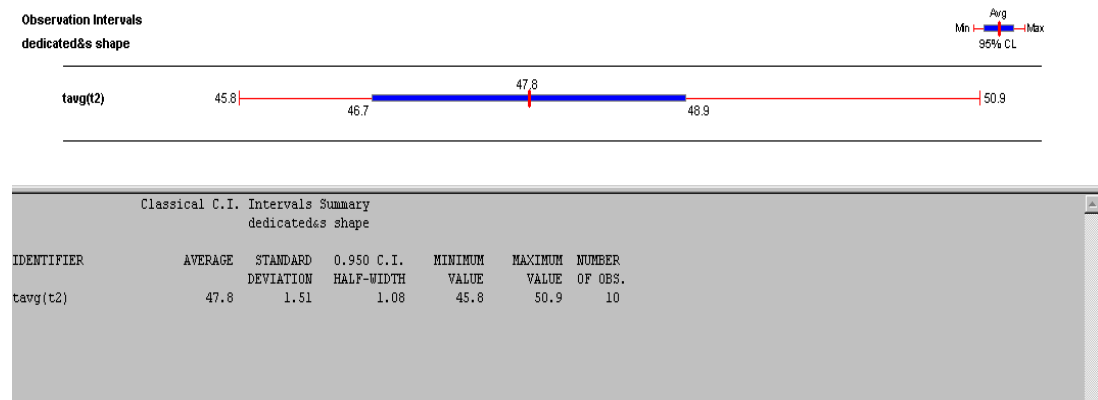


Figure 4.18 95% Confidence interval for mean order retrieval time for system 4

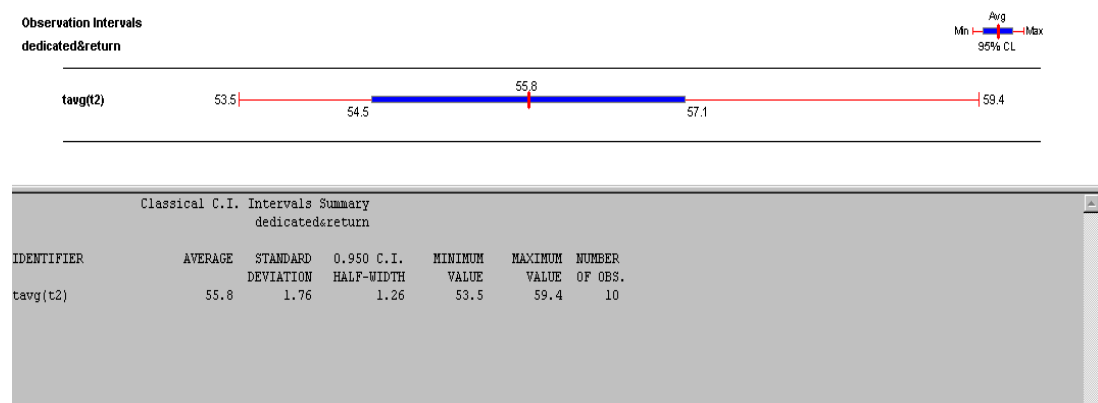


Figure 4.19 95% Confidence interval for mean order retrieval time for system 5

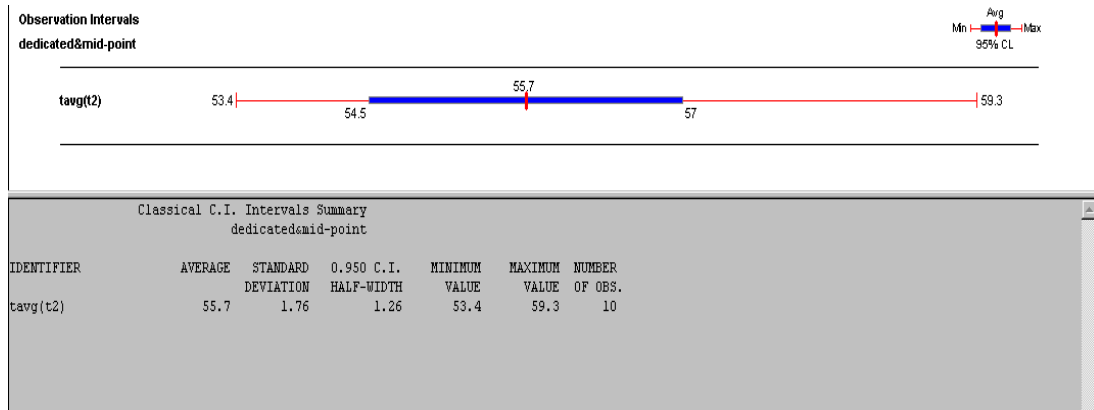


Figure 4.20 95% Confidence interval for mean order retrieval time for system 6

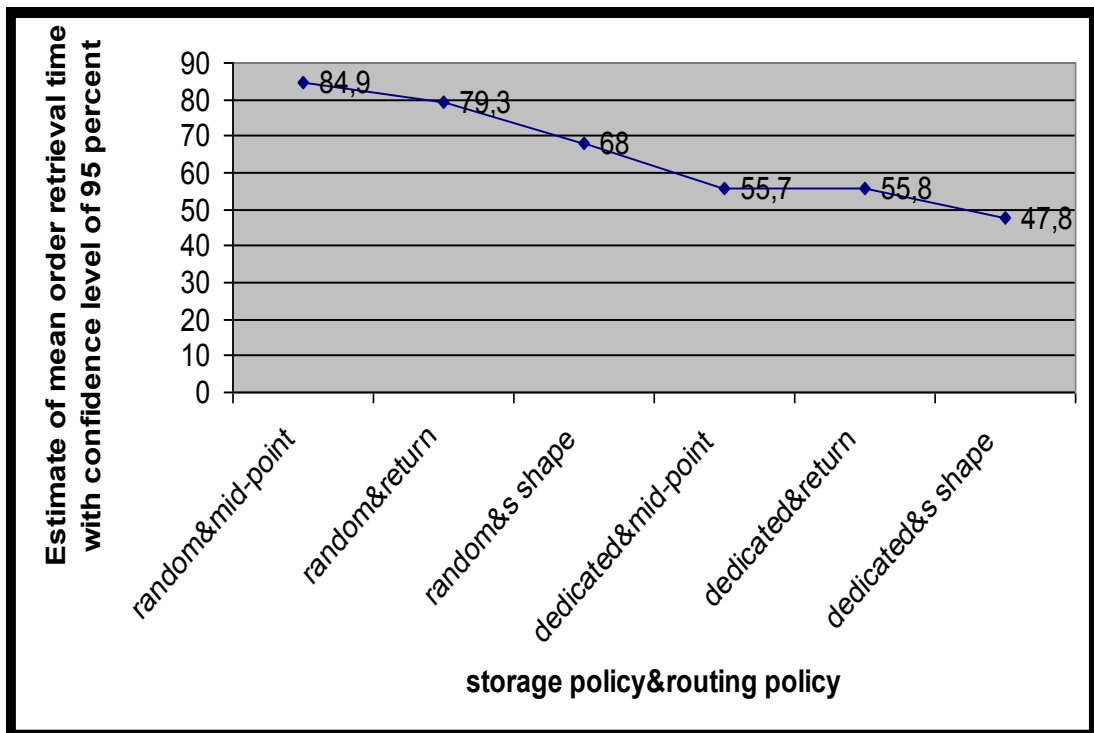


Figure 4.21 Summary of the estimates for the mean order retrieval time respect to different systems

As it can be seen in Figure 4.21 above, minimum order retrieval time has been obtained when employing the dedicated storage policy with S-shaped routing policy. But it is necessary to compare all six systems with each other whether there is a statistically significance respect to mean order retrieval time.

4.3.4 All Pair Wise Comparisons

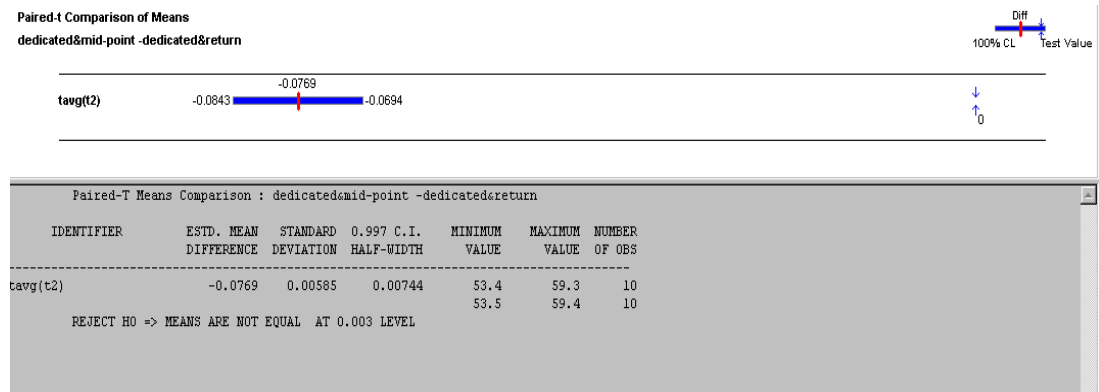
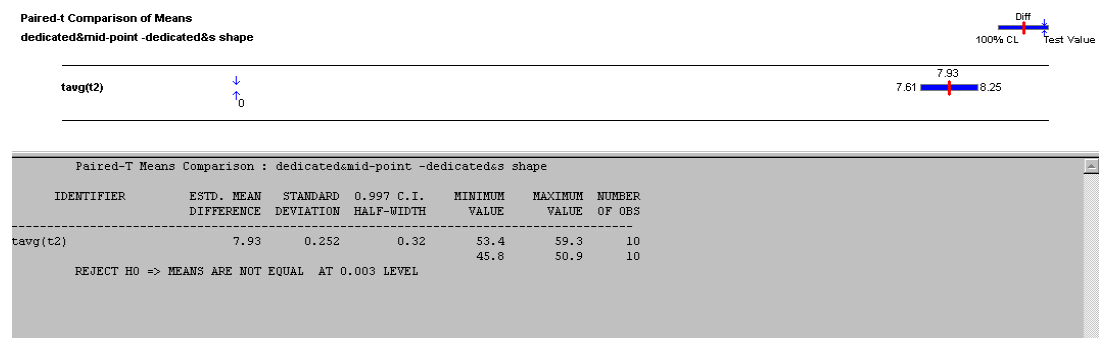
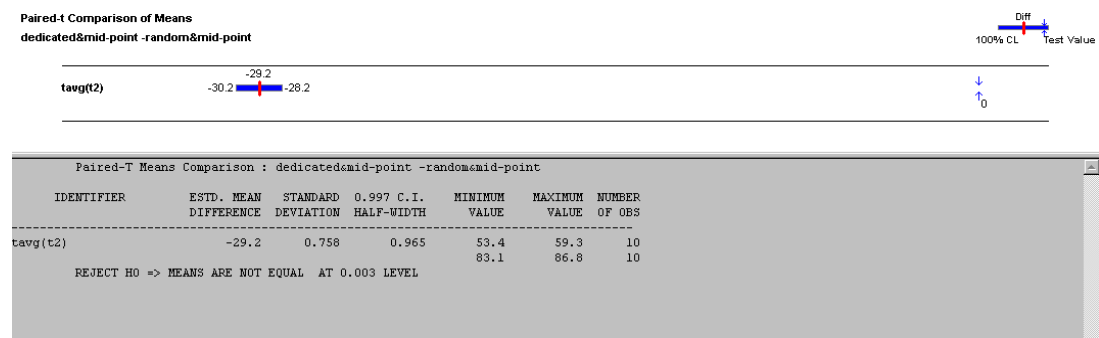
In some studies, we might desire to compare each system with every other system in order to detect and quantify any significant pair wise differences. For example, there may not be an existing system, and all k alternatives represent possible implementations that should be treated in the same way. One approach would be to form confidence intervals for the differences $\mu_{i_1} - \mu_{i_2}$ for all i_1 and i_2 , between 1 and k , with $i_2 < i_1$. With k alternatives, number of confidence intervals computed is $C = k(k - 1)/2$. There will be $k(k - 1)/2$ individual intervals, so depending to the Bonferroni Approach; each must be made at level $1 - \alpha/[k(k - 1)/2]$ in order to have a confidence level of at least $1 - \alpha$ for all the intervals together.

In this section of the study, all order picking system configurations have been compared to each other. Since we have six different systems the basis for comparison is mean order retrieval time μ_i for system i and confidence intervals have been constructed for each comparison. Confidence intervals for $\mu_{i_1} - \mu_{i_2}$ ($i_2 < i_1$) have been constructed having an overall confidence level of 95%. Since the overall error probability is $\alpha = 0.05$ and number of intervals are calculated as $C=15$, individual error probability is $\alpha_c=0.05/15=0.0033$. Since, the simulation models were executed for 10 replications, the degree of freedom is calculated as $\gamma = 10-1 = 9$.

$$H_0: \mu_{i_1} - \mu_{i_2} = 0$$

$$H_1: \mu_{i_1} - \mu_{i_2} \neq 0$$

To test the hypotheses above, using the given parameters and output of the simulation model, paired-t test procedure is employed. Paired-t comparisons obtained from the Arena Output Analyzer are presented in Figures 4.22 through 4.36.

Figure 4.22 Paired-t comparison of means $\mu_6 - \mu_5$ Figure 4.23 Paired-t comparison of means $\mu_6 - \mu_4$ Figure 4.24 Paired-t comparison of means $\mu_6 - \mu_3$

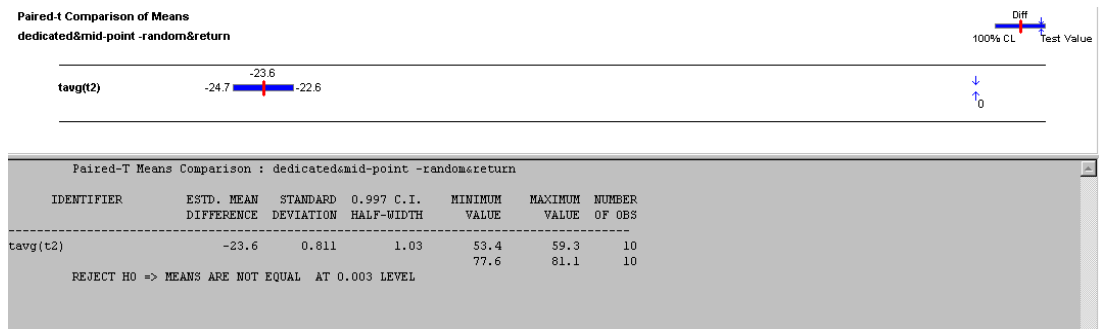


Figure 4.25 Paired-t comparison of means $\mu_6 - \mu_2$

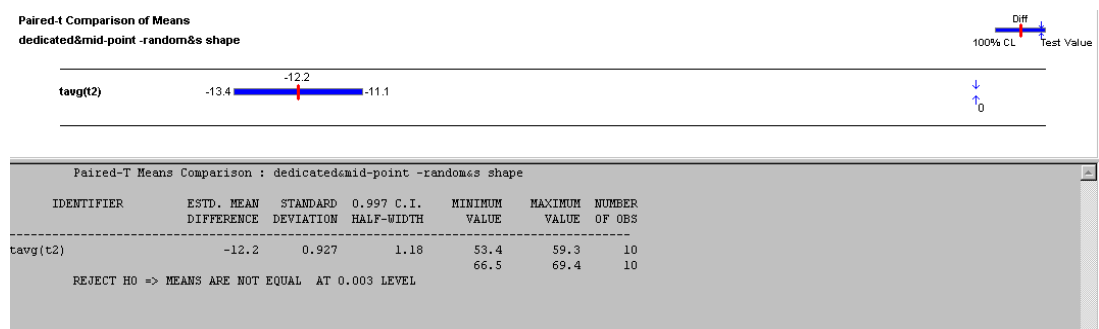


Figure 4.26 Paired-t comparison of means $\mu_6 - \mu_1$

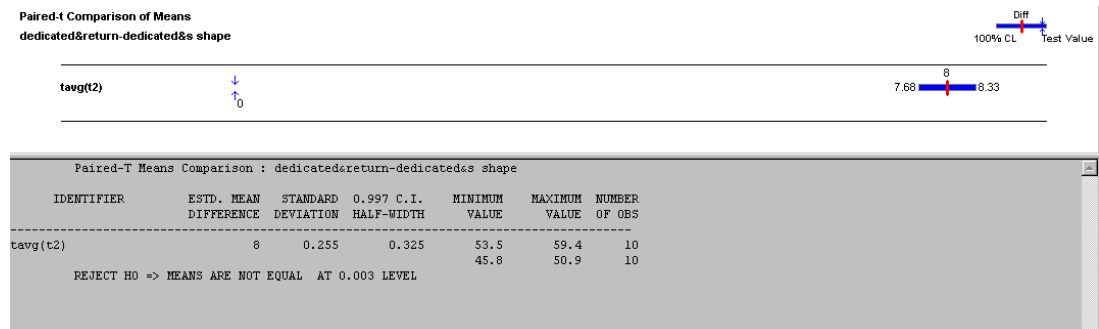
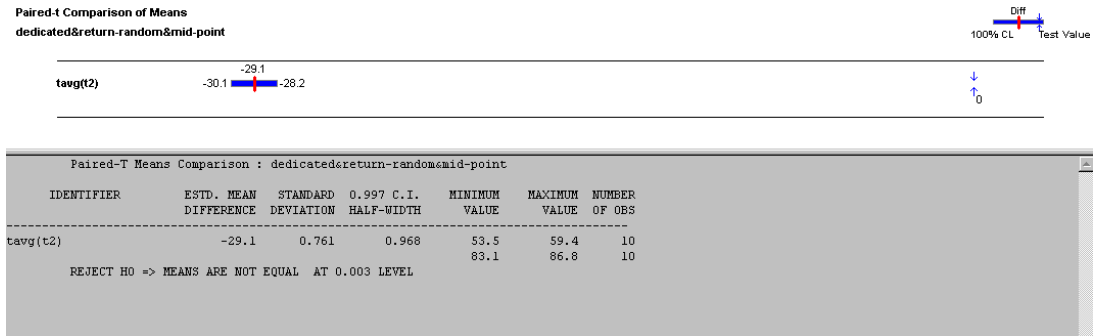
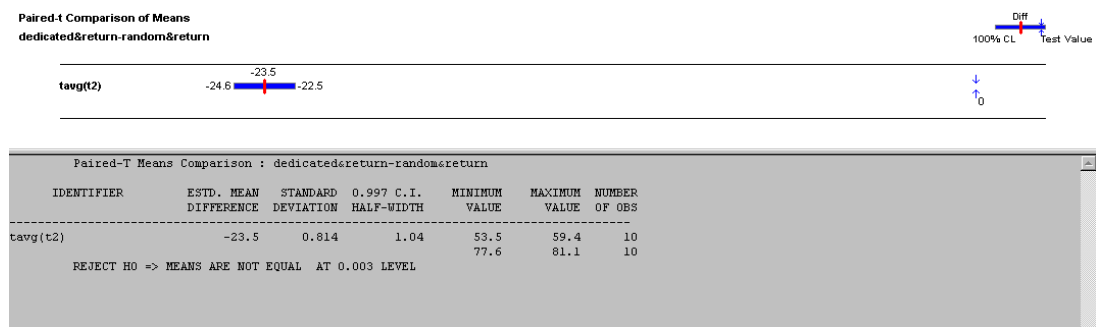
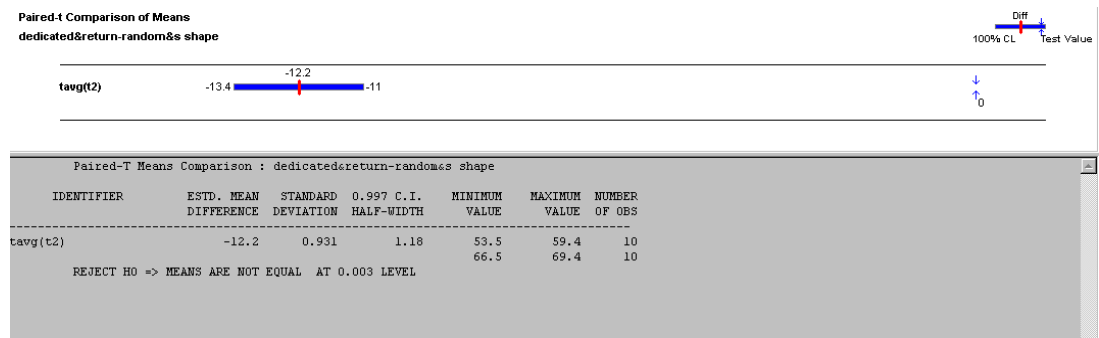
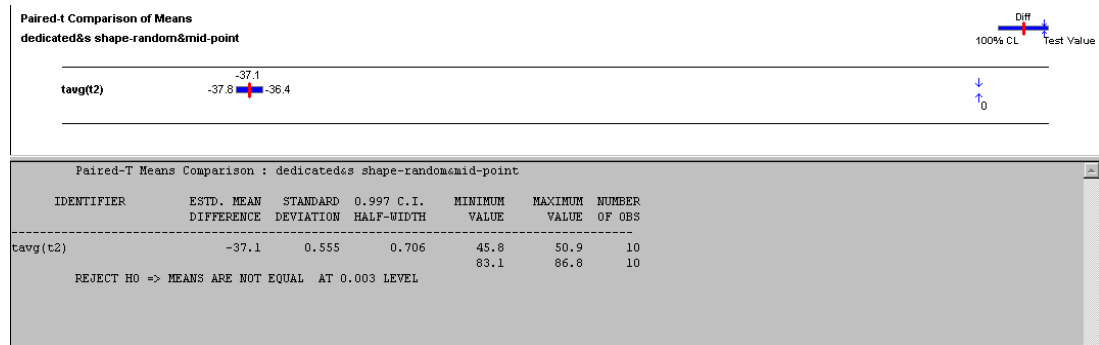
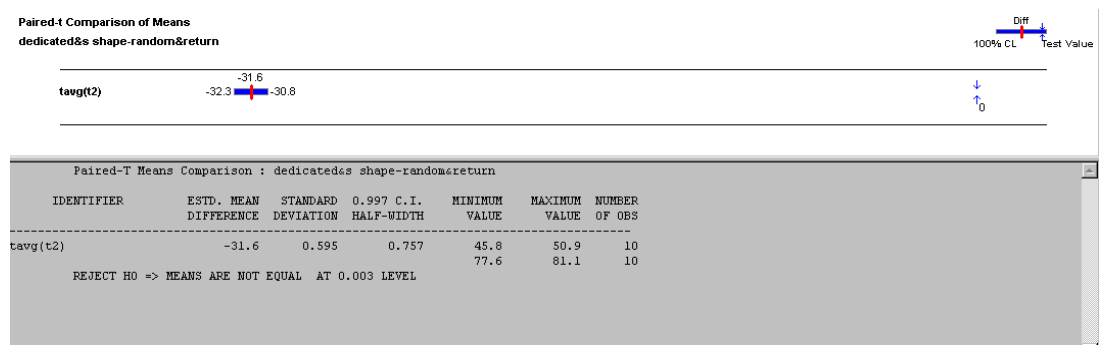
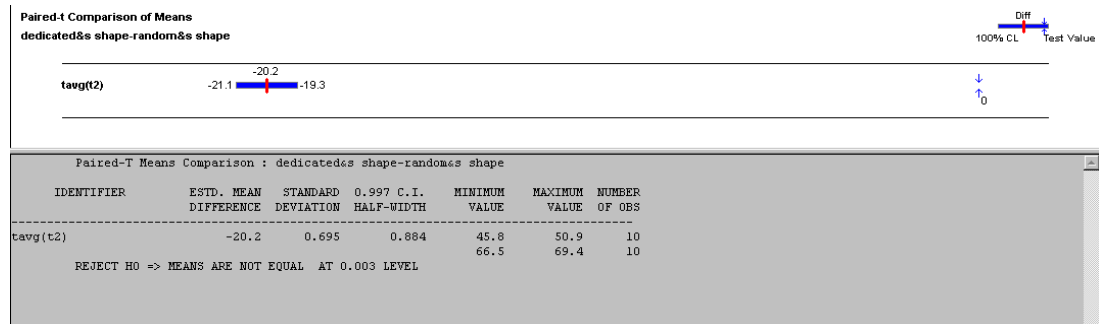


Figure 4.27 Paired-t comparison of means $\mu_5 - \mu_4$

Figure 4.28 Paired-t comparison for means $\mu_5 - \mu_3$ Figure 4.29 Paired-t comparison for means $\mu_5 - \mu_2$ Figure 4.30 Paired-t comparison for means $\mu_5 - \mu_1$

Figure 4.31 Paired-t comparison for means $\mu_4 - \mu_3$ Figure 4.32 Paired-t comparison for means $\mu_4 - \mu_2$ Figure 4.33 Paired-t comparison for means $\mu_4 - \mu_1$

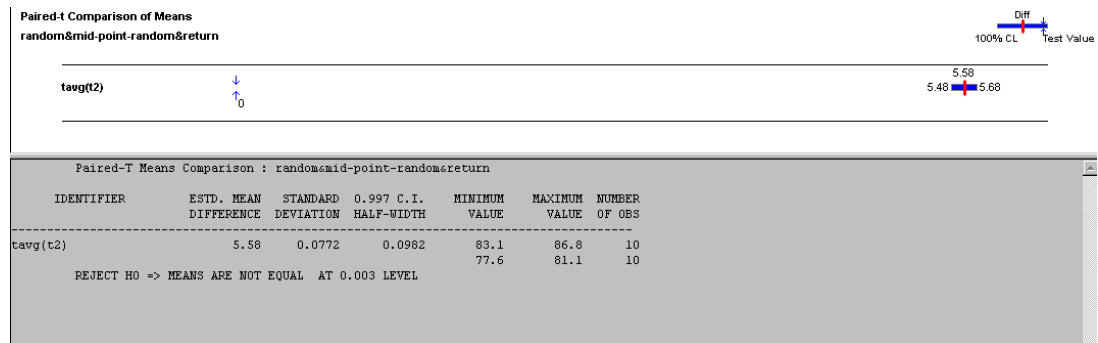
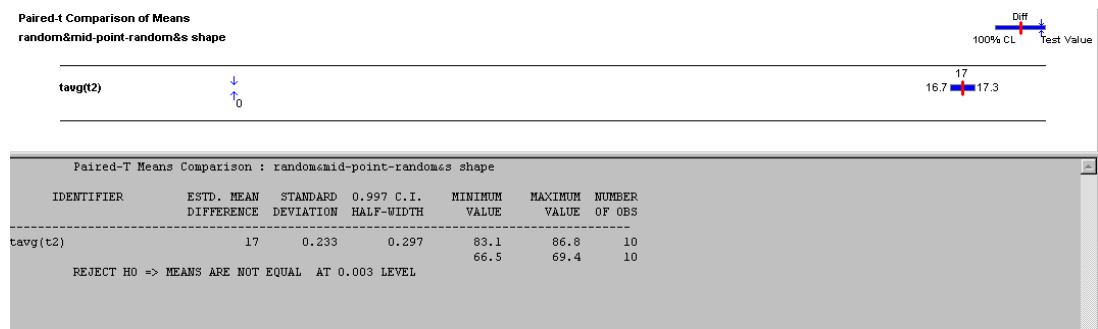
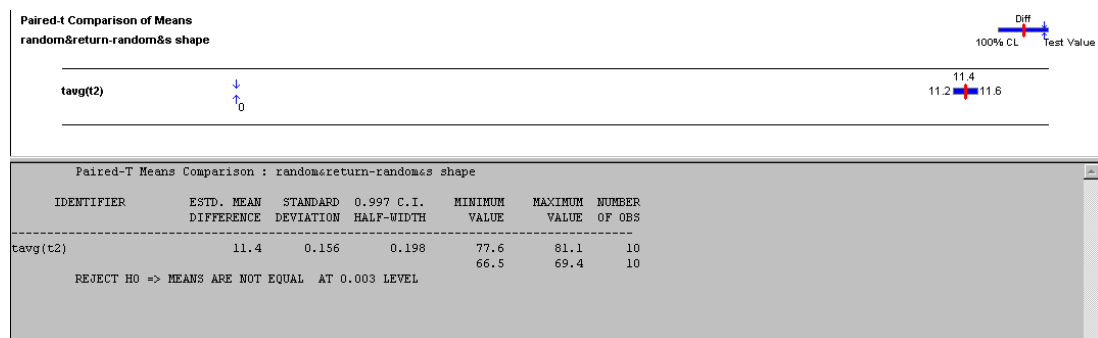
Figure 4.34 Paired-t comparison for means $\mu_3 - \mu_2$ Figure 4.35 Paired-t comparison for means $\mu_3 - \mu_1$ Figure 4.36 Paired-t comparison for means $\mu_2 - \mu_1$

Table 4.1 Summary of the obtained half-width results from the paired-t test

		Paired-t Test					
		System 2					
		1	2	3	4	5	6
System 1	1						
	2	11.4±0.198					
	3	17±0.297	5.58±0.0982				
	4	-20.2±0.884	-31.6±0.757	-37.1±0.706			
	5	-12.2±1.18	-23.5±1.04	-29.1±0.968	8±0.968		
	6	-12.2±1.18	-23.6±1.03	-29.2±0.965	7.93±0.32	-0.0769±0.00744	

When the figures of the analysis are observed, since none of the half widths include zero, H_0 is rejected for each comparison. And the result reveals that there is statistically significant difference between the each pair of the order picking systems.

CHAPTER FIVE

RESULTS AND FUTURE RESEARCH

This thesis aims to provide models as decision-support systems for controlling the efficiency of an order picking system. A hypothetical warehouse is considered and a low level, picker-to-parts order picking system employing discrete picking has been evaluated. In order to obtain the shortest order retrieval time, different combinations of the storage policies (dedicated storage policy, random storage policy) and the routing policies (S-shaped, return and mid-point routing policies) have been applied and compared using the simulation models. Statistical analyses have been carried out on the implementation results. For each system, confidence interval estimate for order retrieval time is depicted. Also, multiple comparisons have been evaluated to determine whether there is a statistically significant difference between the each pair of the order picking systems or not. According to the results, the shortest order retrieval time is obtained when employing the dedicated storage policy and S-shaped routing policy together and there is statistically significance difference between the each pair of the order picking systems.

Different from the most studies performed, replenishment of the order picking system has been considered in this study. Replenishment of the each product type in the warehouse is performed with respect to the inventory policy named (S, s) . Application of this policy turns the order picking system into a dynamic system and makes an outstanding difference among the similar studies performed so far.

The proposed simulation model provides the decision-maker to improve the order picking systems comprehensively. By this way, various aspects of the order picking problem can be evaluated. Since the model is parametric, the model can be executed for various warehouse layouts with different number of aisles, bays and distances. Furthermore, this model can aid the decision-maker in analyzing different performance measures such as total travelled distance, utilization of the racks and the customer satisfaction connected with the amount of the orders which are not met on time.

On the other hand, traffic congestion can be considered, while increasing the number of order pickers. ABC analyses can be carried out and class-based or full turnover storage policies can be evaluated by combining the rest of the other routing policies. Additionally, in the future researches, problem of determining the S , s values for optimal replenishment can be considered.

REFERENCES

- Armstrong R.D., Cook V.D.&Saipé A.L. (1979). Optimal Batching in a Semi-Automated Order Picking System. *The Journal of the Operational Research Society*, 30(8), 711-720.
- Bartholdi J. J.&Hackman S. T. (2005). Warehouse & distribution science. Available on line at: <http://www.tli.gatech.edu/whscience/book/wh-sci.pdf>.
- Benders, J.F. (1962). Partitioning procedures for solving mixed-variable programming problems. *Numerische Mathematic*, 4, 238-252.
- Bindi F., Manzini R., Pareschi A.&Regattieri A. (2009). Similarity-based storage allocation rules in an order picking system: an application to the food service industry. *International Journal of Logistics: Research and Applications*, 12(4), 233-247.
- Bottani, E.&Rizzi, A., (2008). Economical assessment of the impact of RFID technology and EPC system on the Fast Moving Consumer Goods supply chain. *International Journal of Production Economics*, 112 (2), 548–569.
- 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.
- 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.
- Chan F.T.S.&Chan H.K. (2011). Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications*, 38, 2686-2700.

- Chang F.L., Liu Z.X., Xin Z.&Liu D.D. (2007). Research on Order Picking Optimization Problem of Automated Warehouse. *Systems Engineering - Theory & Practice*, 27(2), 139–143.
- Dallari F., Marchet G.&Melacini M. (2009). Design of order picking system. *International Journal of Advanced Manufacturing Technology*, 42(1-2), 1-12.
- De Koster R., 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.
- Goetschalckx M.&Ratliff H.D. (1988). Sequencing picking operations in a man-on-board order picking system. *Material Flow* 4, 255-263.
- Gu J., Goetschalckx M.&McGinnis L.F. (2007). Research on warehouse operation: A comprehensive review. *European Journal of Operational Research*, 177, 1–21.
- Hall R.W., (1993). Distance approximate for routing manual pickers in a warehouse. *IIE Transactions*, 25(4), 76-87.
- Heragu S.S., Du L., Mantel R.J.&Schuur P.B. (2005). Mathematical Model for Warehouse Design and Product Allocation. *International Journal of Production Research*, 43(5), 327-338.
- Heskett J.L. (1964). Putting the cube-per-order index to work in warehouse layout. *Transport and Distribution Management*, 4, 23-30.
- Hsieh L.F.&Huang Y.C., (2011). New batch construction heuristics to optimise the performance of order picking systems. *International Journal of Production Economics*, 131, 618-630.

- Hwang H.S.&Cho G.S., (2006). A performance evaluation model for order picking warehouse design. *Computers and Industrial Engineering*, 51(2), 335-342.
- Le-Duc T.&De Koster R., (2005). Layout optimization for class based storage strategy warehouses. In De Koster R., Delfmann W. (Eds.), *Supply Chain Management – European Perspectives*. CBS Press, Copenhagen, pp. 191–214.
- Lerher T., Potrc I., Sraml M.&Tollazi T., (2010). Travel time models for automated warehouses with aisle transferring storage and retrieval machine. *European Journal of Operational Research*, 205, 571-583.
- Lerher T., Potrc I.&Sraml M., (2011). Simulation analysis of mini-load multi-shuttle automated storage and retrieval systems. *International Journal of Advanced Manufacturing Technology*, 54, 337-348.
- Manzini R., Gamberi M.&Regattieri A. (2005). Design and control of a flexible order picking system (FOPS) a new integrated approach to the implementation of an expert system. *Journal of Manufacturing Technology Management* 16(1), 18–35.
- MHIA (2006). Material Handling Taxonomy, Order Picker. Material Handling Industry of America, Charlotte, NC. Available at <http://www.mhia.org/et/mhetax.htm>.
- Oudijk D., Roodbergen K.J., De Koster R.&Mekern M., (1999). The interactive Erasmus Logistica Warehouse Website. <http://www.fbk.eur.nl/OZ/LOGISTICA/>
- Pan J.C.H.&Shih P.H., (2008). Evaluation of the throughput of a multiple-picker order picking system with congestion consideration. *Computers and Industrial Engineering*, 55(2), 379-389.

- Pan J.C.H.&Wu M.H., (2009). A study of storage assignment problem for an order picking line in a pick-and-pass warehousing system. *Computers and Industrial Engineering*, 57, 261-268.
- Pan J.C.H., Shih P.H.&Wu M.H., (2011). Storage assignment problem with travel distance and blocking considerations for a picker-to-part order picking system. *Computers and Industrial Engineering*, 62, 527-535.
- Parikh P.J.&Meller R.D., (2008). Selecting between batch and zone order picking strategies in a distribution center. *Transportation Research Part E*, 44(5), 696-719.
- Petersen C.G., (1997). An evaluation of order picking routing policies. *International Journal of Operations & Production Management*, 17(11), 1098-1111.
- 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.
- Petersen C.G.&Aase G., (2004). A comparison of picking, storage, and routing policies in manual order picking. *International Journal of Production Economics* 92, 11–19.
- 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.
- Roodbergen K.J.&De Koster R., (1998). Routing order pickers in a warehouse with multiple cross aisles. In R. J. Graves, L. F. McGinnis, D. J. Medeiros, R. E. Ward and M. R. Wilhelm (eds.), *Progress in Material Handling Research: (Charlotte, NC: Material Handling Institute)*, 451-467.

- Roodbergen K.J.&De Koster R., (2001a). Routing methods for warehouses with multiple cross aisles. *International Journal of Production Research*, 39(9), 1865-1883.
- Roodbergen K.J.&De Koster R., (2001b). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research* 133(1), 32-43
- Roodbergen K.J., (2005). Storage assignment policies for warehouses with multiple cross aisles. In: Meller, R., Ogle, M.K., Peters, B.A., Taylor, G.D., Usher, J. (Eds.), *Progress in Material Handling Research*, 541–560.
- Roodbergen K.J.&Vis I.F.A., (2006). A model for warehouse layout. *IIE Transactions* 38(10), 799-811.
- Roodbergen K.J.&Vis I.F.A., (2009). A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, 194, 343-362.
- Rouwenhorst B, Reuter B., Stockrahm V., Van Houtum G.J., Mantel R.J.&Zijm W.H.M., (2000). Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122, 515-533.
- Tompkins J.A., White J.A., Bozer Y.A., Frazelle E.H.&Tanchoco J.M.A., (2003). *Facilities Planning*, (NJ: John Wiley & Sons).
- Tuna G. & Tunçel G. (2012). Depo Yönetiminde Sipariş Toplama Sistemleri: Bir Literatür Araştırması. *DEÜ Mühendislik Bilimleri Dergisi*, Cilt 14, Sayı 3.
- Vaughan T.S.&Petersen C.G., (1999). The effect of warehouse cross aisles on order picking efficiency. *International Journal of Production Research*, 37, 881-897.

Yu, M.&De Koster, R., (2009). The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, 198(2), 480-490.

100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,

100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,

100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100:

j1,CLEAR(System),CATEGORY("None-None"):

j2,CLEAR(System),CATEGORY("None-None");

QUEUES: 1,q1,HighValueFirst(random),,AUTOSTATS(Yes,,):

2,requestq,FirstInFirstOut,,AUTOSTATS(Yes,,):

qgiris,FirstInFirstOut,,AUTOSTATS(Yes,,);

PICTURES: Picture.Report;

RESOURCES:

r1,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(Yes,,),EFFICIENCY(1,
);

STATIONS: 1,giris1,,,,AUTOSTATS(Yes,,):

2,k11,,,,AUTOSTATS(Yes,,):

3,k12,,,,AUTOSTATS(Yes,,):

4,k13,,,,AUTOSTATS(Yes,,):

5,k14,,,,AUTOSTATS(Yes,,):

6,cikis1,,,,AUTOSTATS(Yes,,):

7,giris2,,,,AUTOSTATS(Yes,,):

8,k21,,,,AUTOSTATS(Yes,,):

9,k22,,,,AUTOSTATS(Yes,,):

10,k23,,,,AUTOSTATS(Yes,,):

11,k24,,,,AUTOSTATS(Yes,,):

12,cikis2,,,,AUTOSTATS(Yes,,):

13,giris3,,,,AUTOSTATS(Yes,,):

14,k31,,,,AUTOSTATS(Yes,,):

15,k32,,,,AUTOSTATS(Yes,,):

16,k33,,,,AUTOSTATS(Yes,,):

17,k34,,,,AUTOSTATS(Yes,,):

18,cikis3,,,,AUTOSTATS(Yes,,):

19,giris4,,,,AUTOSTATS(Yes,,):

20,k41,,,,AUTOSTATS(Yes,,):

21,k42,,,,AUTOSTATS(Yes,,):

22,k43,,,,AUTOSTATS(Yes,,):

23,k44,,,,AUTOSTATS(Yes,,):

24,cikis4,,,,AUTOSTATS(Yes,,):

25,depot,,,,AUTOSTATS(Yes,,);

DISTANCES: Distance 1,cikis1-k44-8,cikis1-k43-10,cikis3-k43-6,cikis3-k44-
4,k43-cikis2-8,k44-cikis2-6,cikis4-cikis1-6,depot-

k41-12,depot-k43-16,depot-k44-18,depot-k42-14,depot-giris1-

4,giris1-k11-2,k11-k12-2,k12-k13-2,k13-k14-2,k14-

cikis1-2,cikis1-cikis2-2,cikis2-k24-2,k23-k24-2,k22-k23-2,k21-

k22-2,giris2-k21-2,giris2-giris3-2,giris3-k31-2,k31-

k32-2,k32-k33-2,k34-k33-2,k34-cikis3-2,depot-cikis1-14,depot-

cikis2-16,depot-cikis3-18,depot-k11-6,depot-k12-8,

depot-k13-10,depot-k14-12,depot-giris2-6,depot-k21-8,depot-

k22-10,depot-k23-12,depot-k24-16,depot-giris3-8,depot-

k31-10,depot-k32-12,depot-k33-14,depot-k34-16,giris1-k12-

4,giris1-k13-6,giris1-k14-8,giris1-cikis1-10,giris1-

giris2-2,giris1-k21-4,giris1-k22-6,giris1-k23-8,giris1-k24-

10,giris1-giris3-4,giris1-k31-6,giris1-k32-8,giris1-

k33-10,giris1-k34-12,giris2-k22-4,giris2-k23-6,giris2-k24-

8,giris2-giris3-2,giris2-cikis2-10,giris2-k31-4,giris2-

k32-6,giris2-k34-10,giris2-k33-8,giris2-cikis3-12,giris2-k11-

4,giris2-k12-6,giris2-k14-10,giris2-k13-8,giris2-

```

        cikis1-12,depot-cikis1-14,cikis1-k11-8,cikis1-k12-6,cikis1-
k13-4,cikis1-k14-2,cikis1-k24-4,cikis1-k23-6,cikis1-
        k22-8,cikis1-k21-10,cikis1-cikis3-4,cikis1-k34-6,cikis1-k33-
8,cikis1-k32-10,cikis1-k31-12,cikis2-depot-16,cikis2-
        k24-2,cikis2-k23-4,cikis2-k22-6,cikis2-k21-8,cikis2-giris1-
12,cikis2-k11-10,cikis2-k12-8,cikis2-k13-6,cikis2-k14-
        4,cikis2-cikis3-2,cikis2-k34-4,cikis2-k33-6,cikis2-k32-
8,cikis2-k31-10,cikis3-depot-18,cikis3-k34-2,cikis3-k33-4,
        cikis3-k32-6,cikis3-k31-8,cikis3-cikis2-2,cikis3-k24-4,cikis3-
k23-6,cikis3-k22-8,cikis3-k21-10,cikis3-k14-6,
        cikis3-k13-8,cikis3-k12-10,cikis3-k11-12,giris3-k11-6,giris3-
k12-8,giris3-k13-10,giris3-k14-12,cikis1-giris3-14,
        giris3-k21-4,giris3-k22-6,giris3-k23-8,giris3-k24-10,giris3-
cikis2-12,giris3-k31-2,k32-giris3-4,giris3-k33-6,
        giris3-k34-8,cikis3-giris3-10,k11-k21-6,k11-k22-8,k11-k31-
4,k11-k32-10,k12-k21-8,k12-k22-10,k12-k31-10,k12-k32-12,
        k21-k31-6,k21-k32-8,k14-k24-6,k14-k23-8,k14-k33-10,k14-k34-
8,k13-k24-8,k13-k23-10,k13-k34-10,k13-k33-12,k24-k34-6,
        k24-k33-8,k23-k34-8,k23-k33-10,k22-k31-8,depot-giris4-
10,giris4-k41-2,k41-k42-2,k42-k43-2,k43-k44-2,k44-cikis4-2,
        giris4-k33-8,giris4-k34-10,giris4-k24-12,depot-k44-18,giris4-
k31-4,giris4-k32-6,giris4-k21-6,giris4-k22-8,giris4-
        k23-10,giris4-k11-8,giris4-k12-10,giris4-k13-12,giris4-k14-
14,cikis4-cikis2-4,cikis3-cikis4-2,giris4-giris3-2,
        cikis4-depot-20,cikis3-giris4-12;

```

```

TRANSPORTERS: aracl,1,Distance(1),200---,Station(25)-
Active,AUTOSTATS(Yes,,);

```

```

TALLIES:      t1:
              t2:
              t3:
              t4;

```

```

DSTATS:      toplanacak(1):
              rota(9):
              toplanacak(2):
              toplanacak1(1):
              toplanacak2(1):
              toplanacak(3):
              toplanacak1(2):
              toplanacak2(2):
              toplanacak(4):
              toplanacak1(3):
              toplanacak2(3):
              toplanacak(5):
              toplanacak(6):
              toplanacak(7):
              toplanacak(8):
              rota(1):
              toplanacak(9):
              rota(2):
              satir2(2):
              rota(3):
              t1:
                t2:
              t3:
              rota(4):
              rota(5):
              rota(6):
              rota(7):
              sutun2(2):
              rota(8);

```

```

OUTPUTS:     tavg(t1):

```

```
tavg(t2),"random_midpoint.dat":
tavg(t3);
```

```
REPLICATE,
10,,,Yes,Yes,MinutesToBaseTime(15000),,,24,Minutes,No,No,,,Yes;
```

Mod. File of the Simulation Model (Random Storage)

```
REATE,          1:150,500:MARK(timein):NEXT(25$);

;
;
;   Model statements for module:  Assign 1
;
ASSIGN:         Picture=Picture.Report:
                buyuks=tnow+100000000000000:NEXT(26$);

QUEUE,         qgiris;
SEIZE,         1,Other:
                r1,1:NEXT(41$);

TALLY:         t3,int(timein),1;
DELAY:         0.000000000000000001,,Other:NEXT(43$);
ASSIGN:         sayac=0:
                k2=ANINT(unif(1,tipsayisi,1)):
                c=1;
DUPLICATE:     tipsayisi-1,45$:NEXT(45$);

ASSIGN:         sayac=sayac+1:
                tip=sayac:
                random=unif(1,10,2);
BRANCH,        1:
                If,nq(q1)<>(tipsayisi-1),46$,Yes:
                Else,49$,Yes;
QUEUE,         q1:DETACH;
DUPLICATE:     1,50$:NEXT(46$);

WHILE:         c<=k2;
ASSIGN:         a(c+tipsayisi)=ANINT(unif(1,100,3)):
                a(c)=AQUE(1,c,50):
                c=c+1;
ENDWHILE;
DUPLICATE:     1,53$:NEXT(12$);

PICKUP:        q1,1,nq(q1);
SPLIT::NEXT(52$);

DISPOSE:       No;

ASSIGN:         c=1:
                b=0:
                rotasezgiseli=3:
                deposezgiseli=1;
WHILE:         c<=k2;
BRANCH,        1:
                If,deposezgiseli==1,314$,Yes:
                If,deposezgiseli==2,0$,Yes;
BRANCH,        1:
                If,toplammiktar(cesit(c))-talep(c)<=kucuks,315$,Yes:
                Else,0$,Yes;
ASSIGN:         siparismik=BigS(cesit(c))-toplammiktar(cesit(c));
```

```

        branch1=branch1+1:
        j1=maxsatir:
        j2=1;
WHILE:    j1>0;
WHILE:    j2<=maxsutun;
BRANCH,  1:

If, (mevcut (j1, j2)==0) && (siparisyolda (j1, j2)==0), 328$, Yes:
    Else, 323$, Yes;
ASSIGN:   siparisyolda (j1, j2)=1:
          replenishtime=norm (20, 3);
DELAY:   0.0,, Other:NEXT (321$);

BRANCH,  1:
          If, siparismik>=100, 320$, Yes:
          If, (100>siparismik) && (siparismik>0), 322$, Yes:
          Else, 327$, Yes;
ASSIGN:   siparismik=siparismik-100:
          mevcut (j1, j2)=100:
          miktar (j1, j2)=mevcut (j1, j2):
          siparisyolda (j1, j2)=0:
          type (j1, j2)=cesit (c):
          date (j1, j2)=ANINT (unif (20, 50)):
          toplammiktar (cesit (c))=toplammiktar (cesit (c))+100;
ASSIGN:   j2=j2+1;
ENDWHILE;
ASSIGN:   j1=j1-1:
          j2=1;
ENDWHILE:NEXT (0$);

ASSIGN:   toplammiktar (cesit (c))=toplammiktar (cesit (c))+siparismik:
          mevcut (j1, j2)=siparismik:
          miktar (j1, j2)=mevcut (j1, j2):
          siparismik=0:
          siparisyolda (j1, j2)=0:
          type (j1, j2)=cesit (c):
          date (j1, j2)=tnow;
ASSIGN:   j1=0:NEXT (326$);

ASSIGN:   i=maxsatir:
          k=1:
          mindate=buyuks;
WHILE:    i>0;
WHILE:    k<=maxsutun;
BRANCH,  1:

If, (type (i, k)==cesit (c)) && (date (i, k)<=mindate) && (miktar (i, k)>0), 4$, Yes:
    Else, 5$, Yes;
ASSIGN:   mindate=date (i, k):
          satir=i:
          sutun=k;
ASSIGN:   k=k+1;
ENDWHILE;
ASSIGN:   i=i-1:
          k=1;
ENDWHILE;
BRANCH,  1:
          If, (satir==0) && (sutun==0), 40$, Yes:
          Else, 23$, Yes;
ASSIGN:   karsilanamayan=karsilanamayan+1;
ASSIGN:   c=c+1:
          satir=0:
          sutun=0;
ENDWHILE:NEXT (61$);
QUEUE,   requestq:MARK (timein2);

```

```

REQUEST,      1:arac1(sds),,depot;
TALLY:        t4,int(timein2),1;
BRANCH,       1:
               If,rotasezgiseli==1,56$,Yes:
               If,rotasezgiseli==2,56$,Yes:
               If,rotasezgiseli==3,74$,Yes:
ASSIGN:        k3=0:
               k4=1:
               b=1:
               toplanacakkalem=rotasayisi;
WHILE:         k3<=koridorsayisi-1;
BRANCH,       1:
               If,rotasezgiseli==1,58$,Yes:
               If,rotasezgiseli==2,65$,Yes:
               If,rotasezgiseli==3,75$,Yes:
ASSIGN:        giris=1:
               donus=0:
               cikis=0;
TRANSPORT:    arac1,1+6*k3;

ASSIGN:        toplamtoplanacak=toplanacak1(k3+1)+toplanacak2(k3+1):
               k4=1:
               koridordayim=0:
               koridordanciktim=0;
IF:            toplamtoplanacak<>0;
TRANSPORT:    arac1,6*k3+1;

ASSIGN:        toplamtoplanacak1=toplanacak1(k3+1):
               k4=1:
               koridordayim=0:
               koridordanciktim=0:
               part=1;
IF:            toplamtoplanacak1<>0;
ASSIGN:        yk3=k3;
TRANSPORT:    arac1,6*k3+1;

ASSIGN:        k3=1:
               k4=1:
               b=1:
               toplanacakkalem=rotasayisi:NEXT(64$);

STATION,      1-25;
BRANCH,       1:
               If,rotasezgiseli==1,208$,Yes:
               If,rotasezgiseli==2,125$,Yes:
               If,rotasezgiseli==3,79$,Yes;
BRANCH,       1:

If,(giris==0)&&(donus==0)&&(cikis==0)&&(m==25),184$,Yes:

If,(giris==1)&&(donus==0)&&(cikis==0)&&(toplanacakkalem<>0),149$,Yes:

If,(giris==0)&&(donus==0)&&(cikis==0)&&(toplanacakkalem<>0),178$,Yes:

If,(giris==0)&&(donus==0)&&(cikis==1)&&(toplanacakkalem<>0),164$,Yes:

If,(giris==0)&&(donus==1)&&(cikis==0)&&(toplanacakkalem<>0),180$,Yes:
               Else,209$,Yes;
FREE:         arac1;
RELEASE:      r1,1;
TALLY:        t1,int(timein),1;
TALLY:        t2,int(timein2),1;
DISPOSE:      No;
FINDJ,        1,rotasayisi:

```

```

((maxgoz*k3<rota(j)) &&(rota(j)<=maxgoz+(maxgoz*k3)) || ((maxsutun+(maxgoz*k3)<
rota(j)) &&(rota(j)<=maxsutun+(maxgoz*k3)+maxgoz)) || ((2*maxsutun+(maxgoz*k3)<
rota(j)) &&(rota(j)<=2*maxsutun+(maxgoz*k3)+maxgoz)) || ((3*maxsutun+(maxgoz*k3)
)<rota(j)) &&(rota(j)<=3*maxsutun+(maxgoz*k3)+maxgoz)) || ((4*maxsutun+(maxgoz*k
3)<rota(j)) &&(rota(j)<=4*maxsutun+(maxgoz*k3)+maxgoz)));
IF:          j<>0;
ASSIGN:      donus=0;
              giris=0;
              yk3=k3+1;
              koridordayim=1;
WHILE:       k4<6;
TRANSPORT:   aracl,6*k3+k4+1;

ASSIGN:      b=1;
WHILE:       b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(k4<>4)*k4) &&(((8*k3<rota(b)) &&(rota(b)<=8+8*k3)) || ((33+8*
k3<=rota(b)) &&(rota(b)<=40+8*k3)) || ((65+8*k3<=rota(b)) &&(rota(b)<=72+8*k3)) |
| ((97+8*k3<=rota(b)) &&(rota(b)<=104+8*k3)) || ((129+8*k3<=rota(b)) &&(rota(b)<=
136+8*k3)));
DELAY:       unif(3,5),,Other:NEXT(154$);

ASSIGN:      mevcut(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplancak(b);

              miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b)):
              toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplancak(b);
IF:          mevcut(satir2(b),sutun2(b))==0;
ASSIGN:      kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:      b=b+1;
ENDWHILE;
ASSIGN:      k4=k4+1;
ENDWHILE;
WHILE:       yk3<=koridorsayisi-1;
ASSIGN:      cikis=1:
              donus=0
TRANSPORT:   aracl,6*yk3+6;

FINDJ,       1,rotasayisi;

((8*yk3<rota(j)) &&(rota(j)<=8+8*yk3)) || ((33+8*yk3<=rota(j)) &&(rota(j)<=40+8*
yk3)) || ((65+8*yk3<=rota(j)) &&(rota(j)<=72+8*yk3)) || ((97+8*yk3<=rota(j)) &&(ro
ta(j)<=104+8*yk3)) || ((129+8*yk3<=rota(j)) &&(rota(j)<=136+8*yk3));
IF:          j<>0;
ASSIGN:      donus=1:
              cikis=0:
              b=1:
              k3=yk3+1:
              koridordayim2=1;
WHILE:       k4>1;
TRANSPORT:   aracl,6*yk3+k4-1;

ASSIGN:      b=1;
WHILE:       b<=rotasayisi;
IF:
              (AMOD(rota(b),4)==((k4-2)<>4)*(k4-
2))+((k4==2)*m)) &&(((8*yk3<rota(b)) &&(rota(b)<=8+8*yk3)) || ((33+8*yk3<=rota(b)
) &&(rota(b)<=40+8*yk3)) || ((65+8*yk3<=rota(b)) &&(rota(b)<=72+8*yk3)) || ((97+8

```



```

*yk3<=rota(b) ) && (rota(b) <=104+8*yk3) ) || ( (129+8*yk3<=rota(b) ) && (rota(b) <=136+
8*yk3) ) );
DELAY:          unif(3,5) , , , , Other:NEXT(170$) ;

ASSIGN:         miktar(satir2(b) , sutun2(b) )=mevcut(satir2(b) , sutun2(b) ) -
toplanacak(b) :

mevcut(satir2(b) , sutun2(b) )=miktar(satir2(b) , sutun2(b) ) :
                toplanacakkalem=toplanacakkalem-1:

toplammiktar(toplanacakcesit(b) )=toplammiktar(toplanacakcesit(b) ) -
toplanacak(b) ;
IF:             mevcut(satir2(b) , sutun2(b) )==0;
ASSIGN:         kontrol(satir2(b) , sutun2(b) )=0;
ENDIF;
ENDIF;
ASSIGN:         b=b+1;
ENDWHILE;
ASSIGN:         k4=k4-1;
ENDWHILE;
ENDIF;
IF:             AMOD(m,6) ==1;
ASSIGN:         yk3=koridorsayisi;
ENDIF;
IF:             koridordayim2==0;
ASSIGN:         yk3=yk3+1;
ENDIF;
ASSIGN:         koridordayim2=0;
ENDWHILE;
ENDIF;
IF:             (AMOD(m,6) ==0) && (yk3==koridorsayisi) ;
ASSIGN:         k3=yk3;
ENDIF;
IF:             koridordayim==0;
ASSIGN:         k3=k3+1;
ENDIF;
ASSIGN:         koridordayim=0:NEXT(68$) ;
BRANCH,        1:

If, (donus==0) && (giris==0) && (cikis==0) && (AMOD(m,6) <>0) , 185$, Yes:

If, (donus==1) && (giris==0) && (cikis==0) && (AMOD(m,6) <>1) , 160$, Yes:
    Else, 211$, Yes;
BRANCH,        1:
                If, m==25, 184$, Yes:
                Else, 210$, Yes;
TRANSPORT:     aracl, depot;

ENDWHILE;
BRANCH,        1:
                If, rotasezgiseli==1, 69$, Yes:
                If, rotasezgiseli==2, 71$, Yes:
                If, rotasezgiseli==3, 80$, Yes;
TRANSPORT:     aracl, depot;

BRANCH,        1:
                If, m<>25, 72$, Yes:
                Else, 115$, Yes;
TRANSPORT:     aracl, depot;

FREE:          aracl;
RELEASE:       r1, 1;
TALLY:        t2, int(timein2) , 1:NEXT(24$) ;

BRANCH,        1:

```



```

If,part==4,256$,Yes:
If,part==5,260$,Yes:
If,part==6,295$,Yes:
If,part==7,101$,Yes:
If,part==8,117$,Yes;
BRANCH,
1:
If,

(AMOD(m,duraksayisi)==1)&&(m<>25)&&(koridordayim==0)&&(koridordanciktim==0)&
&(toplamtoplanacak1<>0),
Yes:
If,

(AMOD(m,duraksayisi)<>1)&&(m<>25)&&(toplamtoplanacak1<>0)&&(koridordayim==1)
&&(koridordanciktim==0),
Yes:
If,m==25,233$,Yes;
WHILE:      toplamtoplanacak1<>0;
ASSIGN:     koridordayim=1;
TRANSPORT:  aracl,6*k3+k4+1;

ASSIGN:     b=1;
WHILE:     b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(k4<>4)*k4)&&(((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*
k3<=rota(b))&&(rota(b)<=40+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))|
|((97+8*k3<=rota(b))&&(rota(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=
136+8*k3)));
DELAY:      unif(3,5),,,,Other:NEXT(220$);
ASSIGN:     miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplancak(b):

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
toplancakkalem=toplancakkalem-1:

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplancak(b):
toplamtoplanacak1=(toplamtoplanacak1)-1;
IF:         mevcut(satir2(b),sutun2(b))==0;
ASSIGN:     kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:     b=b+1;
ENDWHILE;
ASSIGN:     k4=k4+1:
y k4=k4;

ENDWHILE;
ENDIF;
ASSIGN:     k3=k3+1:NEXT(68$);

FREE:      aracl;
RELEASE:   r1,1;
TALLY:    t1,int(timein),1;
TALLY:    t2,int(timein2),1;
DISPOSE:  No;

ASSIGN:     b=1;
WHILE:     b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(yk4<>4)*yk4)&&(((8*yk3<rota(b))&&(rota(b)<=8+8*yk3))||((3
3+8*yk3<=rota(b))&&(rota(b)<=40+8*yk3))||((65+8*yk3<=rota(b))&&(rota(b)<=72+
8*yk3))||((97+8*yk3<=rota(b))&&(rota(b)<=104+8*yk3))||((129+8*yk3<=rota(b))&
&(rota(b)<=136+8*yk3)));

```

```

DELAY:          unif (3,5) , , , , Other: NEXT (240$) ;
ASSIGN:         miktar (satir2 (b) , sutun2 (b) ) = mevcut (satir2 (b) , sutun2 (b) ) -
toplanacak (b) :

mevcut (satir2 (b) , sutun2 (b) ) = miktar (satir2 (b) , sutun2 (b) ) :
toplanacak kalem = toplanacak kalem - 1 ;

toplammiktar (toplanacak cesit (b) ) = toplammiktar (toplanacak cesit (b) ) -
toplanacak (b) :
toplam toplanacak 1 = toplam toplanacak 1 - 1 ;
IF:             mevcut (satir2 (b) , sutun2 (b) ) == 0 ;
ASSIGN:         kontrol (satir2 (b) , sutun2 (b) ) = 0 ;
ENDIF;
ENDIF;
ASSIGN:         b = b + 1 ;
ENDWHILE: NEXT (84$) ;
ASSIGN:         yk4 = yk4 + 1 ;
ENDWHILE;
WHILE:          yk3 >= 1 ;
ASSIGN:         toplam toplanacak 2 = toplanacak 2 (yk3) :
koridor dayim = 0 :
koridor danciktim = 0 :
part = 5 :
k4 = 5 ;
toplamt oplanacak 2 <> 0 ;
ASSIGN:         k3 = yk3 - 1 ;
TRANSPORT:     aracl1,6 * yk3 ;

ASSIGN:         part = 4 :
k3 = koridor sayisi - 1 :
k4 = 1 ;
WHILE:          k4 < durak sayisi - 1 ;
TRANSPORT:     aracl1,6 * k3 + k4 + 1 ;

ASSIGN:         b = 1 ;
WHILE:          b <= rotasayisi ;
IF:
                (AMOD (rota (b) , 4) == ((k4 - 1) <> 4) * (k4 -
1) ) && (( (8 * k3 < rota (b) ) && (rota (b) <= 8 + 8 * k3) ) || ( (33 + 8 * k3 <= rota (b) ) && (rota (b) <= 40
+ 8 * k3) ) || ( (65 + 8 * k3 <= rota (b) ) && (rota (b) <= 72 + 8 * k3) ) || ( (97 + 8 * k3 <= rota (b) ) && (rot
a (b) <= 104 + 8 * k3) ) || ( (129 + 8 * k3 <= rota (b) ) && (rota (b) <= 136 + 8 * k3) ) ) ;
DELAY:          unif (3,5) , , , , Other: NEXT (251$) ;
ASSIGN:         miktar (satir2 (b) , sutun2 (b) ) = mevcut (satir2 (b) , sutun2 (b) ) -
toplanacak (b) :

mevcut (satir2 (b) , sutun2 (b) ) = miktar (satir2 (b) , sutun2 (b) ) :
toplanacak kalem = toplanacak kalem - 1 ;

toplammiktar (toplanacak cesit (b) ) = toplammiktar (toplanacak cesit (b) ) -
toplanacak (b) :
toplam toplanacak 1 = (toplamt oplanacak 1) - 1 ;
IF:             mevcut (satir2 (b) , sutun2 (b) ) == 0 ;
ASSIGN:         kontrol (satir2 (b) , sutun2 (b) ) = 0 ;
ENDIF;
ENDIF;
ASSIGN:         b = b + 1 ;
ENDWHILE: NEXT (91$) ;

ASSIGN:         k4 = k4 + 1 ;
ENDWHILE;
ASSIGN:         yk3 = koridor sayisi - 1 : NEXT (93$) ;

BRANCH,        1 :
                If ,

```

```
(AMOD(m,duraksayisi)==0) && (m<>25) && (koridordayim==0) && (koridordanciktim==0) &
&& (toplamtoplanacak2<>0),
```

```
Yes:
```

```
If,
```

```
(AMOD(m,duraksayisi)<>0) && (m<>25) && (toplamtoplanacak2<>0) && (koridordayim==1)
&& (koridordanciktim==0),
```

```
Yes:
```

```
If,m==25,279$,Yes;
```

```
WHILE: toplamtoplanacak2<>0;
```

```
ASSIGN: koridordayim=1;
```

```
TRANSPORT: aracl,6*yk3-(6-k4);
```

```
ASSIGN: b=1;
```

```
WHILE: b<=rotasayisi;
```

```
IF:
```

```
(AMOD(rota(b),4)==((k4-1)<>4)*(k4-
1)) && ((8*k3<rota(b)) && (rota(b)<=8+8*k3)) || ((33+8*k3<=rota(b)) && (rota(b)<=40
+8*k3)) || ((65+8*k3<=rota(b)) && (rota(b)<=72+8*k3)) || ((97+8*k3<=rota(b)) && (rot
a(b)<=104+8*k3)) || ((129+8*k3<=rota(b)) && (rota(b)<=136+8*k3));
```

```
DELAY: unif(3,5),,,Other:NEXT(266$);
```

```
ASSIGN: mevcut(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b);
```

```
miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b)):
toplanacakkalem=toplanacakkalem-1;
```

```
toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b);
```

```
toplamtoplanacak2=toplamtoplanacak2-1;
```

```
IF: mevcut(satir2(b),sutun2(b))==0;
```

```
ASSIGN: kontrol(satir2(b),sutun2(b))=0;
```

```
ENDIF;
```

```
ENDIF;
```

```
ASSIGN: b=b+1;
```

```
ENDWHILE;
```

```
ASSIGN: k4=k4-1;
```

```
yk4=k4;
```

```
ENDWHILE;
```

```
ENDIF;
```

```
ASSIGN: yk3=yk3-1:NEXT(98$);
```

```
FREE: aracl;
```

```
RELEASE: r1,1;
```

```
TALLY: t1,int(timein),1;
```

```
TALLY: t2,int(timein2),1;
```

```
DISPOSE: No;
```

```
ENDWHILE;
```

```
BRANCH, 1:
If,k3==0,109$,Yes:
Else,107$,Yes;
```

```
WHILE: k4>1;
```

```
ASSIGN: part=6;
```

```
TRANSPORT: aracl,6-(6-k4);
```

```
ASSIGN: part=7;
```

```
TRANSPORT: aracl,cikisl;
```

```
BRANCH, 1:
If,m==25,296$,Yes:
Else,291$,Yes;
```

```
FREE: aracl;
```

```

RELEASE:      r1,1;
TALLY:       t1,int(timein),1;
TALLY:       t2,int(timein2),1;
DISPOSE:     No;

ASSIGN:      b=1;
WHILE:       b<=rotasayisi;
IF:
              (AMOD(rota(b),4)==((k4-1)<>4)*(k4-
1))&&(((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*k3<=rota(b))&&(rota(b)<=40
+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))||((97+8*k3<=rota(b))&&(ro
a(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=136+8*k3)));
DELAY:       unif(3,5),,,,Other:NEXT(286$);
ASSIGN:      mevcut(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplancak(b);

miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b)):
              toplancakkalem=toplancakkalem-1;

toplammiktar(toplancakcesit(b))=toplammiktar(toplancakcesit(b))-
toplancak(b):
              toplamtoplancak2=toplamtoplancak2-1;
IF:          mevcut(satir2(b),sutun2(b))==0;
ASSIGN:      kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:      b=b+1;
ENDWHILE:NEXT(99$);

ASSIGN:      k4=k4-1;
ENDWHILE;
TRANSPORT:   aracl,depot;

ASSIGN:      part=8:
              k4=5:
              k3=1:
              yk3=k3-1;
WHILE:       k4>1;
TRANSPORT:   aracl,6*k3-(6-k4);

BRANCH,      1:
              If,m==25,118$,Yes:
              Else,312$,Yes;
FREE:        aracl;
RELEASE:     r1,1;
TALLY:      t1,int(timein),1;
TALLY:      t2,int(timein2),1;
DISPOSE:     No;

IF:          (k4==2)|| (k4==3);
ASSIGN:      b=1;
WHILE:       b<=rotasayisi;
IF:
              (AMOD(rota(b),4)==((k4-1)<>4)*(k4-
1))&&(((8*yk3<rota(b))&&(rota(b)<=8+8*yk3))||((33+8*yk3<=rota(b))&&(rota(b)<
=40+8*yk3))||((65+8*yk3<=rota(b))&&(rota(b)<=72+8*yk3))||((97+8*yk3<=rota(b)
)&&(rota(b)<=104+8*yk3))||((129+8*yk3<=rota(b))&&(rota(b)<=136+8*yk3)));
DELAY:       unif(3,5),,,,Other:NEXT(303$);

ASSIGN:      miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplancak(b);

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
              toplancakkalem=toplancakkalem-1;

```

```

toplammiktar (toplancakcesit (b))=toplammiktar (toplancakcesit (b))-
toplancak (b) :
                                toplamtoplancak1=(toplamtoplancak1)-1;
IF:                            mevcut (satir2 (b), sutun2 (b))=0;
ASSIGN:                         kontrol (satir2 (b), sutun2 (b))=0;
ENDIF;
ENDIF;
ASSIGN:                         b=b+1;
ENDWHILE;
ENDIF:NEXT (104$);
ASSIGN:                         k4=k4-1;
ENDWHILE;
TRANSPORT:                      aracl, depot;

DISPOSE:                        No;

ASSIGN:                         b=b+1;
BRANCH,                         1:
                                If, miktar (satir, sutun) -talep (c) >= 0, 15$, Yes:
                                Else, 18$, Yes;
ASSIGN:                         kalan=0:
                                miktar (satir, sutun) = miktar (satir, sutun) - talep (c) :
                                toplancak (b) = talep (c) :
                                satir2 (b) = satir:
                                sutun2 (b) = sutun;
BRANCH,                         1:
                                If, miktar (satir, sutun) == 0, 17$, Yes:
                                Else, 13$, Yes;
ASSIGN:                         kontrol (satir, sutun) = 0;
ASSIGN:                         istasyonno =
                                (satir==5) * (sutun) + (satir==4) * (sutun+32) + (satir==3) * (sutun+64) + (satir==2) * (s
                                utun+96) + (satir==1) * (sutun+128) :
                                rota (b) = istasyonno:
                                rotasayisi = rotasayisi + 1:
                                toplancakcesit (b) = cesit (c);
ASSIGN:                         d1=1;
WHILE:                         d1 <= koridorsayisi;
ASSIGN:                         d3=d1-1;
IF:
                                (istasyonno == (1 + (8*d3))) || (istasyonno == (2 + (8*d3))) || (istasyonno == (5 + (8*d3)))
                                || (istasyonno == (6 + (8*d3))) || (istasyonno == (33 + (8*d3))) || (istasyonno == (34 + (8*d
                                3))) || (istasyonno == (37 + (8*d3))) || (istasyonno == (38 + (8*d3))) || (istasyonno == (65
                                + (8*d3))) || (istasyonno == (66 + (8*d3))) || (istasyonno == (69 + (8*d3))) || (istasyonno
                                == (70 + (8*d3))) || (istasyonno == (97 + (8*d3))) || (istasyonno == (98 + (8*d3))) || (istas
                                yonno == (101 + (8*d3))) || (istasyonno == (102 + (8*d3))) || (istasyonno == (129 + (8*d3)))
                                || (istasyonno == (130 + (8*d3))) || (istasyonno == (133 + (8*d3))) || (istasyonno == (134 +
                                (8*d3)));
ASSIGN:                         toplancak1 (d1) = toplancak1 (d1) + 1;
ENDIF;
IF:
                                (istasyonno == 3 + 8 * (d1 - 1)) || (istasyonno == 4 + 8 * (d1 -
                                1)) || (istasyonno == 7 + 8 * (d1 - 1)) || (istasyonno == 8 + 8 * (d1 -
                                1)) || (istasyonno == 35 + 8 * (d1 - 1)) || (istasyonno == 36 + 8 * (d1 -
                                1)) || (istasyonno == 39 + 8 * (d1 - 1)) || (istasyonno == 40 + 8 * (d1 -
                                1)) || (istasyonno == 67 + 8 * (d1 - 1)) || (istasyonno == 68 + 8 * (d1 -
                                1)) || (istasyonno == 71 + 8 * (d1 - 1)) || (istasyonno == 72 + 8 * (d1 -
                                1)) || (istasyonno == 99 + 8 * (d1 - 1)) || (istasyonno == 100 + 8 * (d1 -
                                1)) || (istasyonno == 103 + 8 * (d1 - 1)) || (istasyonno == 104 + 8 * (d1 -
                                1)) || (istasyonno == 131 + 8 * (d1 - 1)) || (istasyonno == 132 + 8 * (d1 -
                                1)) || (istasyonno == 135 + 8 * (d1 - 1)) || (istasyonno == 136 + 8 * (d1 - 1));
ASSIGN:                         toplancak2 (d1) = toplancak2 (d1) + 1;
ENDIF;

```

```
ASSIGN:          d1=d1+1;
ENDWHILE;
BRANCH,         1:
                If, kalan<>0, 21$, Yes:
                Else, 10$, Yes;
ASSIGN:         satir=0:
                sutun=0:NEXT(0$);

ASSIGN:         toplanacak(b)=miktar(satir, sutun):
                kalan=talep(c)-miktar(satir, sutun):
                talep(c)=kalan:
                miktar(satir, sutun)=0:
                kontrol(satir, sutun)=0:
                satir2(b)=satir:
                sutun2(b)=sutun:NEXT(13$);
```


Mod. File of the Simulation Model (Dedicated Storage)

```

CREATE,          1:150,500:MARK(timein):NEXT(25$);

Model statements for module: Assign 1
ASSIGN:          Picture=Picture.Report:
                 buyuks=tnow+10000000000000000:NEXT(26$);
QUEUE,          qgiris;
SEIZE,          1,Other:
                 r1,1:NEXT(41$);
TALLY:          t3,int(timein),1;
DELAY:          0.00000000000000001,,Other:NEXT(43$);

ASSIGN:          sayac=0:
                 k2=ANINT(unif(1,tipsayisi,1)):
                 c=1;
DUPLICATE:      tipsayisi-1,45$:NEXT(45$);

ASSIGN:          sayac=sayac+1:
                 tip=sayac:
                 random=unif(1,10,2);
BRANCH,         1:
                 If,nq(q1)<>(tipsayisi-1),46$,Yes:
                 Else,49$,Yes;
QUEUE,          q1:DETACH;
DUPLICATE:      1,50$:NEXT(46$);

WHILE:          c<=k2;
ASSIGN:          a(c+tipsayisi)=ANINT(unif(1,100,3)):
                 a(c)=AQUE(1,c,50):
                 c=c+1;

ENDWHILE;
DUPLICATE:      1,53$:NEXT(12$);
PICKUP:         q1,1,nq(q1);
SPLIT::NEXT(52$);
DISPOSE:        No;

ASSIGN:          c=1:
                 b=0:
                 rotasezgiseli=3:
                 deposezgiseli=2;
WHILE:          c<=k2;
BRANCH,         1:
                 If,deposezgiseli==1,313$,Yes:
                 If,deposezgiseli==2,328$,Yes;
BRANCH,         1:
                 If,toplammiktar(cesit(c))-talep(c)<=kucuks,314$,Yes:
                 Else,0$,Yes;
ASSIGN:          siparismik=BigS(cesit(c))-toplammiktar(cesit(c)):
                 branch1=branch1+1:
                 j1=maxsatir:
                 j2=1;
WHILE:          j1>0;
WHILE:          j2<=maxsutun;
BRANCH,         1:

If,(mevcut(j1,j2)==0)&&(siparisyolda(j1,j2)==0),327$,Yes:
                 Else,322$,Yes;
ASSIGN:          siparisyolda(j1,j2)=1;
DELAY:          0.0,,Other:NEXT(320$);

BRANCH,         1:
                 If,siparismik>=100,319$,Yes:
                 If,(100>siparismik)&&(siparismik>0),321$,Yes:
                 Else,326$,Yes;

```

```

ASSIGN:      siparismik=siparismik-100:
              mevcut(j1,j2)=100:
              miktar(j1,j2)=mevcut(j1,j2):
              siparisyolda(j1,j2)=0:
              type(j1,j2)=cesit(c):
              date(j1,j2)=ANINT(unif(20,50)):
              toplammiktar(cesit(c))=toplammiktar(cesit(c))+100;
ASSIGN:      j2=j2+1;
ENDWHILE;
ASSIGN:      j1=j1-1:
              j2=1;
ENDWHILE:NEXT(0$);

ASSIGN:      toplammiktar(cesit(c))=toplammiktar(cesit(c))+siparismik:
              mevcut(j1,j2)=siparismik:
              miktar(j1,j2)=mevcut(j1,j2):
              siparismik=0:
              siparisyolda(j1,j2)=0:
              type(j1,j2)=cesit(c):
              date(j1,j2)=ANINT(unif(20,50));
ASSIGN:      j1=0:NEXT(325$);

ASSIGN:      i=maxsatir:
              k=1:
              mindate=buyuks;
WHILE:      i>0;
WHILE:      k<=maxsutun;
BRANCH,    1:

If, (type(i,k)==cesit(c)) && (date(i,k)<=mindate) && (miktar(i,k)>0), 4$, Yes:
           Else, 5$, Yes;
ASSIGN:      mindate=date(i,k):
              satir=i:
              sutun=k;
ASSIGN:      k=k+1;
ENDWHILE;
ASSIGN:      i=i-1:
              k=1;
ENDWHILE;
BRANCH,    1:
           If, (satir==0) && (sutun==0), 40$, Yes:
           Else, 23$, Yes;
ASSIGN:      karsilanamayan=karsilanamayan+1;
ASSIGN:      c=c+1:
              satir=0:
              sutun=0;

ENDWHILE:NEXT(61$);

QUEUE,      requestq:MARK(timein2);
REQUEST,    1:arac1(sds),,depot;
BRANCH,    1:
           If, rotasezgiseli==1, 56$, Yes:
           If, rotasezgiseli==2, 56$, Yes:
           If, rotasezgiseli==3, 74$, Yes;
ASSIGN:      k3=0:
              k4=1:
              b=1:
              toplanacakkalem=rotasayisi;
WHILE:      k3<=koridorsayisi-1;
BRANCH,    1:
           If, rotasezgiseli==1, 58$, Yes:
           If, rotasezgiseli==2, 65$, Yes:
           If, rotasezgiseli==3, 75$, Yes;
ASSIGN:      giris=1:

```

```

                                donus=0:
                                cikis=0;
TRANSPORT:                    arac1,1+6*k3;

ASSIGN:                        toplamtoplanacak=toplanacak1 (k3+1)+toplanacak2 (k3+1) :
                                k4=1:
                                koridordayim=0:
                                koridordanciktim=0;
IF:                            toplamtoplanacak<>0;
TRANSPORT:                    arac1,6*k3+1;

ASSIGN:                        toplamtoplanacak1=toplanacak1 (k3+1) :
                                k4=1:
                                koridordayim=0:
                                koridordanciktim=0:
                                part=1;
IF:                            toplamtoplanacak1<>0;
ASSIGN:                        yk3=k3;
TRANSPORT:                    arac1,6*k3+1;

ASSIGN:                        k3=1:
                                k4=1:
                                b=1:
                                toplanacakkalem=rotasayisi:NEXT (64$) ;

STATION,                       1-25;
BRANCH,                         1:
                                If,rotasezgiseli==1,207$,Yes:
                                If,rotasezgiseli==2,124$,Yes:
                                If,rotasezgiseli==3,79$,Yes;
BRANCH,                         1:

If, (giris==0) && (donus==0) && (cikis==0) && (m==25) ,183$,Yes:

If, (giris==1) && (donus==0) && (cikis==0) && (toplanacakkalem<>0) ,148$,Yes:

If, (giris==0) && (donus==0) && (cikis==0) && (toplanacakkalem<>0) ,177$,Yes:

If, (giris==0) && (donus==0) && (cikis==1) && (toplanacakkalem<>0) ,163$,Yes:

If, (giris==0) && (donus==1) && (cikis==0) && (toplanacakkalem<>0) ,179$,Yes:
                                Else,208$,Yes;

FREE:                          arac1;
TALLY:                          t1,int (timein) ,1;
TALLY:                          t2,int (timein2) ,1;
RELEASE:                        r1,1;
DISPOSE:                        No;

FINDJ,                          1,rotasayisi:

( (maxgoz*k3<rota (j)) && (rota (j) <=maxgoz+ (maxgoz*k3)) || ( (maxsutun+ (maxgoz*k3) <
rota (j)) && (rota (j) <=maxsutun+ (maxgoz*k3) +maxgoz) ) || ( (2*maxsutun+ (maxgoz*k3) <
rota (j)) && (rota (j) <=2*maxsutun+ (maxgoz*k3) +maxgoz) ) || ( (3*maxsutun+ (maxgoz*k3
)<rota (j)) && (rota (j) <=3*maxsutun+ (maxgoz*k3) +maxgoz) ) || (4*maxsutun+ (maxgoz*k
3)<rota (j)) && (rota (j) <=4*maxsutun+ (maxgoz*k3) +maxgoz) ) ;
IF:                              j<>0;
ASSIGN:                          donus=0:
                                giris=0:
                                yk3=k3+1:
                                koridordayim=1;
WHILE:                          k4<6;
TRANSPORT:                    arac1,6*k3+k4+1;

ASSIGN:                          b=1;

```

```

WHILE:          b<=rotasayisi;
IF:

  (AMOD(rota(b),4)==(k4<>4)*k4)&&(((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*
k3<=rota(b))&&(rota(b)<=40+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))|
|((97+8*k3<=rota(b))&&(rota(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=
136+8*k3)));
DELAY:          unif(3,5),,,,Other:NEXT(153$);

ASSIGN:          mevcut(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b);

miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b)):
                toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b);
IF:              mevcut(satir2(b),sutun2(b))==0;
ASSIGN:          kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:          b=b+1;
ENDWHILE;
ASSIGN:          k4=k4+1;
ENDWHILE;
WHILE:          yk3<=koridorsayisi-1;
ASSIGN:          cikis=1;
                donus=0;
TRANSPORT:      arac1,6*yk3+6;

FINDJ,          1,rotasayisi:

  ((8*yk3<rota(j))&&(rota(j)<=8+8*yk3))||((33+8*yk3<=rota(j))&&(rota(j)<=40+8*
yk3))||((65+8*yk3<=rota(j))&&(rota(j)<=72+8*yk3))||((97+8*yk3<=rota(j))&&(ro
ta(j)<=104+8*yk3))||((129+8*yk3<=rota(j))&&(rota(j)<=136+8*yk3));
IF:              j<>0;
ASSIGN:          donus=1;
                cikis=0;
                b=1;
                k3=yk3+1;
WHILE:          k4>1;
TRANSPORT:      arac1,6*yk3+k4-1;

ASSIGN:          b=1;
WHILE:          b<=rotasayisi;
IF:

  (AMOD(rota(b),4)==((k4-2)<>4)*(k4-
2))+((k4==2)*m)&&(((8*yk3<rota(b))&&(rota(b)<=8+8*yk3))||((33+8*yk3<=rota(b)
))&&(rota(b)<=40+8*yk3))||((65+8*yk3<=rota(b))&&(rota(b)<=72+8*yk3))||((97+8
*yk3<=rota(b))&&(rota(b)<=104+8*yk3))||((129+8*yk3<=rota(b))&&(rota(b)<=136+
8*yk3)));
DELAY:          unif(3,5),,,,Other:NEXT(169$);

ASSIGN:          miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b);

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
                toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b);
IF:              mevcut(satir2(b),sutun2(b))==0;
ASSIGN:          kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;

```

```

ASSIGN:          b=b+1;
ENDWHILE;
ASSIGN:          k4=k4-1;
ENDWHILE;
ENDIF;
IF:              AMOD(m,6)==1;
ASSIGN:          yk3=koridorsayisi;
ENDIF;
IF:              koridordayim2==0;
ASSIGN:          yk3=yk3+1;
ENDIF;
ASSIGN:          koridordayim2=0;
ENDWHILE;
ENDIF;
IF:              (AMOD(m,6)==0) && (yk3==koridorsayisi);
ASSIGN:          k3=yk3;
ENDIF;
IF:              koridordayim==0;
ASSIGN:          k3=k3+1;
ENDIF;
ASSIGN:          koridordayim=0:NEXT(68$);

BRANCH,          1:

If, (donus==0) && (giris==0) && (cikis==0) && (AMOD(m,6) <>0), 184$, Yes:

If, (donus==1) && (giris==0) && (cikis==0) && (AMOD(m,6) <>1), 159$, Yes:
    Else, 210$, Yes;

BRANCH,          1:
    If, m==25, 183$, Yes:
    Else, 209$, Yes;
TRANSPORT:      aracl, depot;

ENDWHILE;
BRANCH,          1:
    If, rotasezgiseli==1, 69$, Yes:
    If, rotasezgiseli==2, 71$, Yes:
    If, rotasezgiseli==3, 80$, Yes;
TRANSPORT:      aracl, depot;

BRANCH,          1:
    If, m <> 25, 72$, Yes:
    Else, 115$, Yes;
TRANSPORT:      aracl, depot;

FREE:           aracl;
TALLY:          t2, int(timein2), 1;
RELEASE:        r1, 1:NEXT(24$);

BRANCH,          1:
    If, yk3==koridorsayisi-1, 82$, Yes:
    Else, 86$, Yes;
WHILE:          yk4 < duraksayisi-1;
ASSIGN:         part=2;
TRANSPORT:      aracl, 6*yk3+yk4+1;

ASSIGN:         part=3;
                k4=1;
TRANSPORT:      aracl, 6*(koridorsayisi-1)+1;

BRANCH,          1:
    If,

```

(AMOD(m,duraksayisi)==1) && (m <> 25) && (koridordayim==0) && (koridordanciktim==0) && (toplantoplanacak <> 0),

```

Yes:
If,

(AMOD(m,duraksayisi)<>1)&&(m<>25)&&(toplamtoplanacak<>0)&&(koridordayim==1)&
&(koridordanciktim==0),
Yes:
If,m==25,143$,Yes;
WHILE:      toplamtoplanacak<>0;
ASSIGN:     koridordayim=1;
TRANSPORT:  aracl,6*k3+k4+1;

ASSIGN:     b=1;
WHILE:     b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(k4<>4)*k4)&&(((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*
k3<=rota(b))&&(rota(b)<=40+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))|
|((97+8*k3<=rota(b))&&(rota(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=
136+8*k3)));
DELAY:      unif(3,5),,,,Other:NEXT(130$);

ASSIGN:     miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplancak(b);

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
toplancakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplancak(b):
toplamtoplanacak=toplamtoplancak-1;
IF:         mevcut(satir2(b),sutun2(b))==0;
ASSIGN:     kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:     b=b+1;
ENDWHILE;
ASSIGN:     k4=k4+1;
ENDWHILE;
ENDIF;
ASSIGN:     k3=k3+1:NEXT(68$);

FREE:      aracl;
TALLY:     t1,int(timein),1;
TALLY:     t2,int(timein2),1;
RELEASE:   r1,1;
DISPOSE:   No;
BRANCH,    1:
If,part==1,213$,Yes:
If,part==2,244$,Yes:
If,part==3,88$,Yes:
If,part==4,255$,Yes:
If,part==5,259$,Yes:
If,part==6,294$,Yes:
If,part==7,101$,Yes:
If,part==8,117$,Yes;
BRANCH,    1:
If,

(AMOD(m,duraksayisi)==1)&&(m<>25)&&(koridordayim==0)&&(koridordanciktim==0)&
&(toplamtoplanacak1<>0),
214$,Yes:
If,

(AMOD(m,duraksayisi)<>1)&&(m<>25)&&(toplamtoplanacak1<>0)&&(koridordayim==1)
&&(koridordanciktim==0),

```

```

Yes:
                                If,m==25,232$,Yes;
WHILE:      toplamtoplanacak1<>0;
ASSIGN:     koridordayim=1;
TRANSPORT:  aracl,6*k3+k4+1;

ASSIGN:     b=1;
WHILE:     b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(k4<>4)*k4)&&((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*
k3<=rota(b))&&(rota(b)<=40+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))|
|((97+8*k3<=rota(b))&&(rota(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=
136+8*k3));
DELAY:     unif(3,5),,,,Other:NEXT(219$);

ASSIGN:     miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b):

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
                                toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b):
                                toplamtoplanacak1=(toplamtoplanacak1)-1;
IF:     mevcut(satir2(b),sutun2(b))==0;
ASSIGN: kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN: b=b+1;
ENDWHILE;
ASSIGN: k4=k4+1;
                                yk4=k4;
ENDWHILE;
ENDIF;
ASSIGN: k3=k3+1:NEXT(68$);

FREE:     aracl;
TALLY:   t1,int(timein),1;
TALLY:   t2,int(timein2),1;
RELEASE: r1,1;
DISPOSE: No;

ASSIGN:   b=1;
WHILE:   b<=rotasayisi;
IF:

(AMOD(rota(b),4)==(yk4<>4)*yk4)&&((8*yk3<rota(b))&&(rota(b)<=8+8*yk3))||((3
3+8*yk3<=rota(b))&&(rota(b)<=40+8*yk3))||((65+8*yk3<=rota(b))&&(rota(b)<=72+
8*yk3))||((97+8*yk3<=rota(b))&&(rota(b)<=104+8*yk3))||((129+8*yk3<=rota(b))&
&(rota(b)<=136+8*yk3));
DELAY:   unif(3,5),,,,Other:NEXT(239$);

ASSIGN:   miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b):

mevcut(satir2(b),sutun2(b))=miktar(satir2(b),sutun2(b)):
                                toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b):
                                toplamtoplanacak1=toplamtoplanacak1-1;
IF:     mevcut(satir2(b),sutun2(b))==0;
ASSIGN: kontrol(satir2(b),sutun2(b))=0;
ENDIF;

```

```

ENDIF;
ASSIGN:      b=b+1;
ENDWHILE:NEXT (84$);

ASSIGN:      yk4=yk4+1;
ENDWHILE;
WHILE:      yk3>=1;
ASSIGN:      toplamtoplanacak2=toplanacak2 (yk3) :
koridordayim=0:
koridordanciktim=0:
part=5:
k4=5;
IF:         toplamtoplanacak2<>0;
ASSIGN:      k3=yk3-1;
TRANSPORT:   aracl,6*yk3;

ASSIGN:      part=4:
k3=koridorsayisi-1:
k4=1;
WHILE:      k4<duraksayisi-1;
TRANSPORT:   aracl,6*k3+k4+1;

ASSIGN:      b=1;
WHILE:      b<=rotasayisi;
IF:
(AMOD (rota (b) , 4) == ( (k4-1) <>4) * (k4-
1) ) && ( ( (8*k3<rota (b) ) && (rota (b) <=8+8*k3) ) || ( (33+8*k3<=rota (b) ) && (rota (b) <=40
+8*k3) ) || ( (65+8*k3<=rota (b) ) && (rota (b) <=72+8*k3) ) || ( (97+8*k3<=rota (b) ) && (rota
(b) <=104+8*k3) ) || ( (129+8*k3<=rota (b) ) && (rota (b) <=136+8*k3) ) ) );
DELAY:      unif (3,5) , , , , Other:NEXT (250$) ;
ASSIGN:      miktar (satir2 (b) , sutun2 (b) ) =mevcut (satir2 (b) , sutun2 (b) ) -
toplanacak (b) :

mevcut (satir2 (b) , sutun2 (b) ) =miktar (satir2 (b) , sutun2 (b) ) :
toplanacakkalem=toplanacakkalem-1:

toplammiktar (toplanacakcesit (b) ) =toplammiktar (toplanacakcesit (b) ) -
toplanacak (b) :
toplamtoplanacak1=(toplamtoplanacak1) -1;
IF:         mevcut (satir2 (b) , sutun2 (b) ) ==0;
ASSIGN:      kontrol (satir2 (b) , sutun2 (b) ) =0;
ENDIF;
ENDIF;
ASSIGN:      b=b+1;
ENDWHILE:NEXT (91$);

ASSIGN:      k4=k4+1;
ENDWHILE;
ASSIGN:      yk3=koridorsayisi-1:NEXT (93$);
BRANCH,     1:
If,
(AMOD (m, duraksayisi) ==0) && (m<>25) && (koridordayim==0) && (koridordanciktim==0) &
& (toplamtoplanacak2<>0) ,
260$, Yes:
If,
(AMOD (m, duraksayisi) <>0) && (m<>25) && (toplamtoplanacak2<>0) && (koridordayim==1)
&& (koridordanciktim==0) ,
270$, Yes:
If, m==25, 278$, Yes;
WHILE:      toplamtoplanacak2<>0;
ASSIGN:      koridordayim=1;
TRANSPORT:   aracl,6*yk3- (6-k4) ;

```



```

ASSIGN:          b=1;
WHILE:           b<=rotasayisi;
IF:
                (AMOD(rota(b),4)==(k4-1)<>4)*(k4-
1) )&&((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*k3<=rota(b))&&(rota(b)<=40
+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))||((97+8*k3<=rota(b))&&(rot
a(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=136+8*k3)));
DELAY:          unif(3,5),,,Other:NEXT(265$);

ASSIGN:          mevcut(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b))-
toplanacak(b);

miktar(satir2(b),sutun2(b))=mevcut(satir2(b),sutun2(b)):
                toplanacakkalem=toplanacakkalem-1;

toplammiktar(toplanacakcesit(b))=toplammiktar(toplanacakcesit(b))-
toplanacak(b):
                toplanacak2=toplanacak2-1;
IF:             mevcut(satir2(b),sutun2(b))==0;
ASSIGN:         kontrol(satir2(b),sutun2(b))=0;
ENDIF;
ENDIF;
ASSIGN:         b=b+1;
ENDWHILE;
ASSIGN:         k4=k4-1;
                yk4=k4;
ENDWHILE;
ENDIF;
ASSIGN:         yk3=yk3-1:NEXT(98$);

FREE:          aracl;
TALLY:         t1,int(timein),1;
TALLY:         t2,int(timein2),1;
RELEASE:       r1,1;
DISPOSE:       No;

ENDWHILE;
BRANCH,        1:
                If,k3==0,109$,Yes:
                Else,107$,Yes;

WHILE:         k4>1;
ASSIGN:         part=6;
TRANSPORT:     aracl,6-(6-k4);

ASSIGN:         part=7;
TRANSPORT:     aracl,cikis1;

BRANCH,        1:
                If,m==25,295$,Yes:
                Else,290$,Yes;

FREE:          aracl;
TALLY:         t1,int(timein),1;
TALLY:         t2,int(timein2),1;
RELEASE:       r1,1;
DISPOSE:       No;

ASSIGN:         b=1;
WHILE:         b<=rotasayisi;
IF:
                (AMOD(rota(b),4)==(k4-1)<>4)*(k4-
1) )&&((8*k3<rota(b))&&(rota(b)<=8+8*k3))||((33+8*k3<=rota(b))&&(rota(b)<=40
+8*k3))||((65+8*k3<=rota(b))&&(rota(b)<=72+8*k3))||((97+8*k3<=rota(b))&&(rot
a(b)<=104+8*k3))||((129+8*k3<=rota(b))&&(rota(b)<=136+8*k3)));
DELAY:         unif(3,5),,,Other:NEXT(285$);

```

```

ASSIGN:      mevcut (satir2 (b) , sutun2 (b) )=mevcut (satir2 (b) , sutun2 (b) ) -
toplanacak (b) :

miktar (satir2 (b) , sutun2 (b) )=mevcut (satir2 (b) , sutun2 (b) ) :
                toplanacakkalem=toplanacakkalem-1:

toplammiktar (toplanacakcesit (b) )=toplammiktar (toplanacakcesit (b) ) -
toplanacak (b) :
                toplamtoplanacak2=toplamtoplanacak2-1;
IF:          mevcut (satir2 (b) , sutun2 (b) )==0;
ASSIGN:      kontrol (satir2 (b) , sutun2 (b) )=0;
ENDIF;
ENDIF;
ASSIGN:      b=b+1;
ENDWHILE: NEXT (99$) ;

9ASSIGN:     k4=k4-1;
ENDWHILE;
TRANSPORT:   aracl, depot;

ASSIGN:      part=8:
                k4=5:
                k3=1:
                yk3=k3-1;
WHILE:       k4>1;
TRANSPORT:   aracl, 6*k3- (6-k4) ;

BRANCH,      1:
                If, m==25, 118$, Yes:
                Else, 311$, Yes;
FREE:        aracl;
TALLY:       t1, int (timein) , 1;
TALLY:       t2, int (timein2) , 1;
RELEASE:     r1, 1;
DISPOSE:     No;
IF:          (k4==2) || (k4==3) ;ASSIGN:      b=1;
WHILE:       b<=rotasayisi;
IF:          (AMOD (rota (b) , 4) == ( (k4-1) <>4) * (k4-
1) ) && ( (8*yk3<rota (b) ) && (rota (b) <=8+8*yk3) ) || ( (33+8*yk3<=rota (b) ) && (rota (b) <
=40+8*yk3) ) || ( (65+8*yk3<=rota (b) ) && (rota (b) <=72+8*yk3) ) || ( (97+8*yk3<=rota (b)
) && (rota (b) <=104+8*yk3) ) || ( (129+8*yk3<=rota (b) ) && (rota (b) <=136+8*yk3) ) ) ;
DELAY:       unif (3, 5) , , , , Other: NEXT (302$) ;
ASSIGN:      miktar (satir2 (b) , sutun2 (b) )=mevcut (satir2 (b) , sutun2 (b) ) -
toplanacak (b) :

mevcut (satir2 (b) , sutun2 (b) )=miktar (satir2 (b) , sutun2 (b) ) :
                toplanacakkalem=toplanacakkalem-1:

toplammiktar (toplanacakcesit (b) )=toplammiktar (toplanacakcesit (b) ) -
toplanacak (b) :
                toplamtoplanacak1=(toplamtoplanacak1) -1;
IF:          mevcut (satir2 (b) , sutun2 (b) )==0;
ASSIGN:      kontrol (satir2 (b) , sutun2 (b) )=0;
ENDIF;
ENDIF;
ASSIGN:      b=b+1;
ENDWHILE;
ENDIF: NEXT (104$) ;

ASSIGN:      k4=k4-1;
ENDWHILE;
TRANSPORT:   aracl, depot;

DISPOSE:     No;

```

```

ASSIGN:      b=b+1;
BRANCH,      1:
                If, miktar (satir, sutun) - talep (c) >= 0, 15$, Yes:
                Else, 18$, Yes;
ASSIGN:      kalan=0:
                miktar (satir, sutun) = miktar (satir, sutun) -
talep (c) :
                toplanacak (b) = talep (c) :
                satir2 (b) = satir:
                sutun2 (b) = sutun;
BRANCH,      1:
                If, miktar (satir, sutun) == 0, 17$, Yes:
                Else, 13$, Yes;
ASSIGN:      kontrol (satir, sutun) = 0;
ASSIGN:      istasyonno=
(satir==5) * (sutun) + (satir==4) * (sutun+32) + (satir==3) * (sutun+64) + (satir==2) * (s
utun+96) + (satir==1) * (sutun+128) :
                rota (b) = istasyonno:
                rotasayisi = rotasayisi + 1:
                toplanacakcesit (b) = cesit (c) ;
ASSIGN:      d1=1;
WHILE:      d1 <= koridorsayisi;
ASSIGN:      d3=d1-1;
IF:
(istasyonno==(1+(8*d3))) || (istasyonno==(2+(8*d3))) || (istasyonno==(5+(8*d3)))
|| (istasyonno==(6+(8*d3))) || (istasyonno==(33+(8*d3))) || (istasyonno==(34+(8*d
3))) || (istasyonno==(37+(8*d3))) || (istasyonno==(38+(8*d3))) || (istasyonno==(65
+(8*d3))) || (istasyonno==(66+(8*d3))) || (istasyonno==(69+(8*d3))) || (istasyon
==(70+(8*d3))) || (istasyonno==(97+(8*d3))) || (istasyonno==(98+(8*d3))) || (istas
yonno==(101+(8*d3))) || (istasyonno==(102+(8*d3))) || (istasyonno==(129+(8*d3)))
|| (istasyonno==(130+(8*d3))) || (istasyonno==(133+(8*d3))) || (istasyonno==(134+
(8*d3)));
ASSIGN:      toplanacak1 (d1) = toplanacak1 (d1) + 1;
ENDIF;
IF:
                (istasyonno==3+8*(d1-1)) || (istasyonno==4+8*(d1-
1)) || (istasyonno==7+8*(d1-1)) || (istasyonno==8+8*(d1-
1)) || (istasyonno==35+8*(d1-1)) || (istasyonno==36+8*(d1-
1)) || (istasyonno==39+8*(d1-1)) || (istasyonno==40+8*(d1-
1)) || (istasyonno==67+8*(d1-1)) || (istasyonno==68+8*(d1-
1)) || (istasyonno==71+8*(d1-1)) || (istasyonno==72+8*(d1-
1)) || (istasyonno==99+8*(d1-1)) || (istasyonno==100+8*(d1-
1)) || (istasyonno==103+8*(d1-1)) || (istasyonno==104+8*(d1-
1)) || (istasyonno==131+8*(d1-1)) || (istasyonno==132+8*(d1-
1)) || (istasyonno==135+8*(d1-1)) || (istasyonno==136+8*(d1-1));
ASSIGN:      toplanacak2 (d1) = toplanacak2 (d1) + 1;
ENDIF;
ASSIGN:      d1=d1+1;
ENDWHILE;
BRANCH,      1:
                If, kalan <> 0, 21$, Yes:
                Else, 10$, Yes;
ASSIGN:      satir=0:
                sutun=0:NEXT (0$);
ASSIGN:      toplanacak (b) = miktar (satir, sutun) :
                kalan = talep (c) - miktar (satir, sutun) :
                talep (c) = kalan:
                miktar (satir, sutun) = 0:
                kontrol (satir, sutun) = 0:
                satir2 (b) = satir:
                sutun2 (b) = sutun:NEXT (13$);

```

```

BRANCH,          1:
                    If, toplammiktar(cesit(c))-
talep(c) <= kucuks, 329$, Yes:
                    Else, 0$, Yes;
ASSIGN:          siparismik=BigS(cesit(c))-toplammiktar(cesit(c)):
                    branch1=branch1+1:
                    j1=

(cesit(c)==1)*5+(cesit(c)==2)*4+(cesit(c)==3)*3+(cesit(c)==4)*2+(cesit(c)==5)
)*1+(cesit(c)==6)*5+(cesit(c)==7)*4+(cesit(c)==8)*3+(cesit(c)==9)*2+(cesit(c)
)==10)*1+(cesit(c)==11)*5+(cesit(c)==12)*4+(cesit(c)==13)*3+(cesit(c)==14)*2
+(cesit(c)==15)*1+(cesit(c)==16)*5+(cesit(c)==17)*4+(cesit(c)==18)*3+(cesit(
c)==19)*2+(cesit(c)==20)*1:
                    j2=

(cesit(c)==1)*1+(cesit(c)==2)*1+(cesit(c)==3)*1+(cesit(c)==4)*1+(cesit(c)==5)
)*1+(cesit(c)==6)*9+(cesit(c)==7)*9+(cesit(c)==8)*9+(cesit(c)==9)*9+(cesit(c)
)==10)*9+(cesit(c)==11)*17+(cesit(c)==12)*17+(cesit(c)==13)*17+(cesit(c)==14)
)*17+(cesit(c)==15)*17+(cesit(c)==16)*25+(cesit(c)==17)*25+(cesit(c)==18)*25
+(cesit(c)==19)*25+(cesit(c)==20)*25;
WHILE:

j2 <= (cesit(c)==1)*4+(cesit(c)==2)*4+(cesit(c)==3)*4+(cesit(c)==4)*4+(cesit(c)
)==5)*4+(cesit(c)==6)*12+(cesit(c)==7)*12+(cesit(c)==8)*12+(cesit(c)==9)*12+
(cesit(c)==10)*12+(cesit(c)==11)*20+(cesit(c)==12)*20+(cesit(c)==13)*20+(ces
it(c)==14)*20+(cesit(c)==15)*20+(cesit(c)==16)*28+(cesit(c)==17)*28+(cesit(c)
)==18)*28+(cesit(c)==19)*28+(cesit(c)==20)*28;
BRANCH,          1:

If, (mevcut(j1,j2)==0) && (siparisyolda(j1,j2)==0), 338$, Yes:
                    Else, 336$, Yes;
ASSIGN:          siparisyolda(j1,j2)=1;
DELAY:           0.0,, Other:NEXT(334$);

BRANCH,          1:
                    If, siparismik >= 100, 333$, Yes:
                    If, (100 > siparismik) && (siparismik > 0), 335$, Yes:
                    Else, 336$, Yes;
ASSIGN:          siparismik=siparismik-100:
                    mevcut(j1,j2)=100:
                    miktar(j1,j2)=mevcut(j1,j2):
                    siparisyolda(j1,j2)=0:
                    type(j1,j2)=cesit(c):
                    date(j1,j2)=tnow:

toplammiktar(cesit(c))=toplammiktar(cesit(c))+100;
ASSIGN:          j2=j2+1;
ENDWHILE:NEXT(339$);

ASSIGN:          siparismik=BigS(cesit(c))-toplammiktar(cesit(c)):
                    branch1=branch1+1:
                    j1=

(cesit(c)==1)*1+(cesit(c)==2)*2+(cesit(c)==3)*3+(cesit(c)==4)*4+(cesit(c)==5)
)*5+(cesit(c)==6)*1+(cesit(c)==7)*2+(cesit(c)==8)*3+(cesit(c)==9)*4+(cesit(c)
)==10)*5+(cesit(c)==11)*1+(cesit(c)==12)*2+(cesit(c)==13)*3+(cesit(c)==14)*4
+(cesit(c)==15)*5+(cesit(c)==16)*1+(cesit(c)==17)*2+(cesit(c)==18)*3+(cesit(
c)==19)*4+(cesit(c)==20)*5:
                    j2=

(cesit(c)==1)*5+(cesit(c)==2)*5+(cesit(c)==3)*5+(cesit(c)==4)*5+(cesit(c)==5)
)*5+(cesit(c)==6)*13+(cesit(c)==7)*13+(cesit(c)==8)*13+(cesit(c)==9)*13+(ces
it(c)==10)*13+(cesit(c)==11)*21+(cesit(c)==12)*21+(cesit(c)==13)*21+(cesit(c)

```

```

) ==14) *21+(cesit(c)==15) *21+(cesit(c)==16) *29+(cesit(c)==17) *29+(cesit(c)==1
8) *29+(cesit(c)==19) *29+(cesit(c)==20) *29;
WHILE:

j2<=(cesit(c)==1) *8+(cesit(c)==2) *8+(cesit(c)==3) *8+(cesit(c)==4) *8+(cesit(c)
)==5) *8+(cesit(c)==6) *16+(cesit(c)==7) *16+(cesit(c)==8) *16+(cesit(c)==9) *16+
(cesit(c)==10) *16+(cesit(c)==11) *24+(cesit(c)==12) *24+(cesit(c)==13) *24+(ces
it(c)==14) *24+(cesit(c)==15) *24+(cesit(c)==16) *32+(cesit(c)==17) *32+(cesit(c)
)==18) *32+(cesit(c)==19) *32+(cesit(c)==20) *32;
BRANCH,      1:

If, (mevcut(j1,j2)==0) && (siparisyolda(j1,j2)==0) , 348$, Yes:
      Else, 346$, Yes;
ASSIGN:      siparisyolda(j1,j2)=1;
DELAY:      0.0,, Other:NEXT(344$);

BRANCH,      1:
      If, siparismik>=100, 343$, Yes:
      If, (100>siparismik) && (siparismik>0) , 345$, Yes:
      Else, 346$, Yes;
ASSIGN:      siparismik=siparismik-100:
      mevcut(j1,j2)=100:
      miktar(j1,j2)=mevcut(j1,j2):
      siparisyolda(j1,j2)=0:
      type(j1,j2)=cesit(c):
      date(j1,j2)=ANINT(unif(50,100)):
      toplammiktar(cesit(c))=toplammiktar(cesit(c))+100;
ASSIGN:      j2=j2+1;
ENDWHILE:NEXT(0$);

ASSIGN:      toplammiktar(cesit(c))=toplammiktar(cesit(c))+siparismik:
      mevcut(j1,j2)=siparismik:
      miktar(j1,j2)=mevcut(j1,j2):
      siparismik=0:
      siparisyolda(j1,j2)=0:
      type(j1,j2)=cesit(c):
      date(j1,j2)=ANINT(unif(20,50)):NEXT(346$);

ASSIGN:      toplammiktar(cesit(c))=toplammiktar(cesit(c))+siparismik:
      mevcut(j1,j2)=siparismik:
      miktar(j1,j2)=mevcut(j1,j2):
      siparismik=0:
      siparisyolda(j1,j2)=0:
      type(j1,j2)=cesit(c):
      date(j1,j2)=ANINT(unif(20,50)):NEXT(336$);

```



```

100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,100,
100,100,100,100,100,100,100,100,100,100,
    100,100,100,100,100,100,100,100,100,100,100,100:
j1,CLEAR(System),CATEGORY("None-None"):
j2,CLEAR(System),CATEGORY("None-None");

QUEUES:    1,q1,HighValueFirst(random),,AUTOSTATS(Yes,,):
           2,requestq,FirstInFirstOut,,AUTOSTATS(Yes,,):
           qgiris,FirstInFirstOut,,AUTOSTATS(Yes,,);

PICTURES:    Picture.Report;

RESOURCES:
r1,Capacity(1),,Stationary,COST(0.0,0.0,0.0),,AUTOSTATS(Yes,,),EFFICIENCY(1,
);

STATIONS:    1,giris1,,,,AUTOSTATS(Yes,,):
           2,k11,,,,AUTOSTATS(Yes,,):
           3,k12,,,,AUTOSTATS(Yes,,):
           4,k13,,,,AUTOSTATS(Yes,,):
           5,k14,,,,AUTOSTATS(Yes,,):
           6,cikis1,,,,AUTOSTATS(Yes,,):
           7,giris2,,,,AUTOSTATS(Yes,,):
           8,k21,,,,AUTOSTATS(Yes,,):
           9,k22,,,,AUTOSTATS(Yes,,):
           10,k23,,,,AUTOSTATS(Yes,,):
           11,k24,,,,AUTOSTATS(Yes,,):
           12,cikis2,,,,AUTOSTATS(Yes,,):
           13,giris3,,,,AUTOSTATS(Yes,,):
           14,k31,,,,AUTOSTATS(Yes,,):
           15,k32,,,,AUTOSTATS(Yes,,):
           16,k33,,,,AUTOSTATS(Yes,,):
           17,k34,,,,AUTOSTATS(Yes,,):
           18,cikis3,,,,AUTOSTATS(Yes,,):
           19,giris4,,,,AUTOSTATS(Yes,,):
           20,k41,,,,AUTOSTATS(Yes,,):
           21,k42,,,,AUTOSTATS(Yes,,):
           22,k43,,,,AUTOSTATS(Yes,,):
           23,k44,,,,AUTOSTATS(Yes,,):
           24,cikis4,,,,AUTOSTATS(Yes,,):
           25,depot,,,,AUTOSTATS(Yes,,);

DISTANCES:    Distance 1,cikis1-k44-8,cikis1-k43-10,cikis3-k43-6,cikis3-k44-
4,k43-cikis2-8,k44-cikis2-6,cikis4-cikis1-6,depot-
    k41-12,depot-k43-16,depot-k44-18,depot-k42-14,depot-giris1-
4,giris1-k11-2,k11-k12-2,k12-k13-2,k13-k14-2,k14-
    cikis1-2,cikis1-cikis2-2,cikis2-k24-2,k23-k24-2,k22-k23-2,k21-
k22-2,giris2-k21-2,giris2-giris3-2,giris3-k31-2,k31-
    k32-2,k32-k33-2,k34-k33-2,k34-cikis3-2,depot-cikis1-14,depot-
cikis2-16,depot-cikis3-18,depot-k11-6,depot-k12-8,
    depot-k13-10,depot-k14-12,depot-giris2-6,depot-k21-8,depot-
k22-10,depot-k23-12,depot-k24-16,depot-giris3-8,depot-
    k31-10,depot-k32-12,depot-k33-14,depot-k34-16,giris1-k12-
4,giris1-k13-6,giris1-k14-8,giris1-cikis1-10,giris1-
    giris2-2,giris1-k21-4,giris1-k22-6,giris1-k23-8,giris1-k24-
10,giris1-giris3-4,giris1-k31-6,giris1-k32-8,giris1-
    k33-10,giris1-k34-12,giris2-k22-4,giris2-k23-6,giris2-k24-
8,giris2-giris3-2,giris2-cikis2-10,giris2-k31-4,giris2-
    k32-6,giris2-k34-10,giris2-k33-8,giris2-cikis3-12,giris2-k11-
4,giris2-k12-6,giris2-k14-10,giris2-k13-8,giris2-
    cikis1-12,depot-cikis1-14,cikis1-k11-8,cikis1-k12-6,cikis1-
k13-4,cikis1-k14-2,cikis1-k24-4,cikis1-k23-6,cikis1-
    k22-8,cikis1-k21-10,cikis1-cikis3-4,cikis1-k34-6,cikis1-k33-
8,cikis1-k32-10,cikis1-k31-12,cikis2-depot-16,cikis2-

```



```

k24-2,cikis2-k23-4,cikis2-k22-6,cikis2-k21-8,cikis2-giris1-
12,cikis2-k11-10,cikis2-k12-8,cikis2-k13-6,cikis2-k14-
4,cikis2-cikis3-2,cikis2-k34-4,cikis2-k33-6,cikis2-k32-
8,cikis2-k31-10,cikis3-depot-18,cikis3-k34-2,cikis3-k33-4,
cikis3-k32-6,cikis3-k31-8,cikis3-cikis2-2,cikis3-k24-4,cikis3-
k23-6,cikis3-k22-8,cikis3-k21-10,cikis3-k14-6,
cikis3-k13-8,cikis3-k12-10,cikis3-k11-12,giris3-k11-6,giris3-
k12-8,giris3-k13-10,giris3-k14-12,cikis1-giris3-14,
giris3-k21-4,giris3-k22-6,giris3-k23-8,giris3-k24-10,giris3-
cikis2-12,giris3-k31-2,k32-giris3-4,giris3-k33-6,
giris3-k34-8,cikis3-giris3-10,k11-k21-6,k11-k22-8,k11-k31-
4,k11-k32-10,k12-k21-8,k12-k22-10,k12-k31-10,k12-k32-12,
k21-k31-6,k21-k32-8,k14-k24-6,k14-k23-8,k14-k33-10,k14-k34-
8,k13-k24-8,k13-k23-10,k13-k34-10,k13-k33-12,k24-k34-6,
k24-k33-8,k23-k34-8,k23-k33-10,k22-k31-8,depot-giris4-
10,giris4-k41-2,k41-k42-2,k42-k43-2,k43-k44-2,k44-cikis4-2,
giris4-k33-8,giris4-k34-10,giris4-k24-12,depot-k44-18,giris4-
k31-4,giris4-k32-6,giris4-k21-6,giris4-k22-8,giris4-
k23-10,giris4-k11-8,giris4-k12-10,giris4-k13-12,giris4-k14-
14,cikis4-cikis2-4,cikis3-cikis4-2,giris4-giris3-2,
cikis4-depot-20,cikis3-giris4-12;

```

```

TRANSPORTERS: aracl,1,Distance (1),200---,Station (25)-
Active,AUTOSTATS (Yes,,);

```

```

TALLIES:      t1:
              t2:
              t3;

```

```

DSTATS:      toplanacak (1) :
              rota (9) :
              toplanacak (2) :
              toplanacak1 (1) :
              toplanacak2 (1) :
              toplanacak (3) :
              toplanacak1 (2) :
              toplanacak2 (2) :
              toplanacak (4) :
              toplanacak1 (3) :
              toplanacak2 (3) :
              toplanacak (5) :
              toplanacak (6) :
              toplanacak (7) :
              toplanacak (8) :
              rota (1) :
              toplanacak (9) :
              rota (2) :
              satir2 (2) :
              rota (3) :
              t1:
              t2:
              t3:
              rota (4) :
              rota (5) :
              rota (6) :
              rota (7) :
              sutun2 (2) :
              rota (8) ;

```

```

OUTPUTS:     tavg (t1) :
              tavg (t2) ,"dedicated_midpoint.dat":
              tavg (t3) ;

```

```

REPLICATE,
10,,,Yes,Yes,MinutesToBaseTime (15000),,,24,Minutes,No,No,,,Yes;

```