ALGORITHMS FOR DYNAMIC FORWARD AREA ALLOCATION IN A WAREHOUSE

by

ŞİLAN HUN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2003

APPROVED BY:

Dr. Gürdal Ertek …….

(Thesis Advisor)

Dr. Tonguç Ünlüyurt …….

Dr. Berrin Yanıkoğlu

…….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Gürdal Ertek for his encouragement, guidance, financial support and help for learning Java.

I would like to thank graduate committee members of my thesis, Dr. Tonguç Ünlüyurt and Dr. Berrin Yanıkoğlu for their critical suggestions and continuous guidance throughout this study.

I would also express my sincere gratitude to Dr. Ersin Özuğurlu and Dr. Kemal Kılıç for their advice, time and insight.

It is obvious that without strong friendship, emotional support, encouragement, and entertainment my graduate study in Sabancı University would be much harder. Therefore, I would like to thank my entire friends and especially my office-mates, Bilge Küçük, Evren Burcu Kıvanç and Özkan Öztürk. I will always remember the enjoyable days spent with Bilge as flat mates. Finally, I want to thank Mehmet Kayhan who makes our life cheerful and amusing while writing our thesis.

I am grateful to N.Mehmet Gökhan for spending time to arrange the order data with Excel. I am also grateful to all the graduate students and faculty members for providing a peaceful environment in the department.

I wish to thank my entire extended family for providing a loving environment for me. With their guidance, love and encouragement, I have overcome many formidable challenges throughout my life.

ABSTRACT

This thesis consists of two major parts. The first part is the presentation of two dynamic algorithms and the second part is the performance analysis of these algorithms using a real order data.

In this study, two algorithms, Slot-based Dynamic Algorithm and Order-based Dynamic Algorithm are developed for dynamic forward area allocation. Both algorithms are based on the idea of reassignment of most profitable items to the forward pick area. For Slot-based Algorithm, the profitable item is selected whenever slot becomes empty. This decision criterion is changed in Order-based Algorithm and becomes the end of definite order cycle. This thesis attempts to determine savings gained by implementing two algorithms in various warehouse settings. Instead of savings, number of forward picks, replenishments between reserve and forward areas and finally number of stock outs are analyzed as performance measures.

Earlier research focused mainly on the relocation of items only in the forward area according to the changing item popularity. On the other hand, our study addresses the issue of reassignment of items from reserve to forward slots in a volatile market environment. Additionally, the forward area is assumed to be a physical space consisting of slots, which makes our proposed algorithms convenient for real life applications.

Our study provides interesting insights in order to increase warehouse efficiency. We observe the advantages and disadvantages of both algorithms in different warehouse settings. We believe that these insights help managers to select the robust methods for creating forward area allocations.

ÖZET

Bu tez başlıca iki önemli kısımdan oluşmaktadır. Birinci kısımda iki dinamik algoritma, ikinci kısımda ise dinamik algoritmaların gerçek sipariş bilgilerinin kullanılması ile elde edilen performanslarının analizleri sunulmaktadır.

Bu çalışmada, depo ön alanının dinamik ataması için Bölme-odaklı ve Sipariş-odaklı olmak üzere iki algoritma geliştirilmiştir. İki algoritma da en fazla kazanç getirecek sipariş mallarının ön depo alanına atanması fikrine dayanır. Bölme odaklı algoritmada herhangi bir bölme boşaldığında, en fazla kazancı getirebilecek olan mal boşalan bölmeye atanır. Bu karar kriteri Sipariş-odaklı algoritmada değişir ve daha önceden tanımlanan sipariş devrinin sonu olarak belirlenir. Bu tez çalışmasında, iki algoritma farklı depo tasarımları için meydana getirdikleri kazançlar bakımından incelenmiştir. Kazanç yanında ön depo alanından yapılan sipariş mallarını toplama sayısı, arka depo alanından ön depo alanına yapılan mal ikmal sayısı ve talep edilen sipariş mallarının ön depo alanında yeteri kadar bulundurulamaması durumları performans kriterleri olarak kullanılabilir.

Geçmiş çalışmalar genellikle ön depo alanındaki malların yerleşiminde değişen müşteri taleplerine göre meydana gelen dinamik değişimleri ele alır. Bizim çalışmamız ise malların arka depo alanından ön depo alanına değişen taleplere göre atanması ile ilgilenir. Ek olarak, ön depo alanının fiziksel bölmelerden oluştuğu düşünülürse, bu yaklaşım ile geliştirdiğimiz algoritmaların gerçek yaşamda uygulanabilirliği de kabul edilebilir.

Çalışmamız depo kullanım etkinliğinin artırılması konusunda ilginç öngörüler sağlamaktadır. Her iki algoritmanın da avantaj ve dezavantajları farklı depo tasarımlar için gözlenmiştir. Bu öngörülerin yöneticilere doğru ön depo atama metotlarını seçebilmeleri için yardımcı olacağına inanıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ALGORITHMS FOR DYNAMIC FORWARD AREA ALLOCATION IN A WAREHOUSE

by

ŞİLAN HUN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
August 2003

APPROVED BY:

Dr. Gürdal Ertek …….

(Thesis Advisor)

Dr. Tonguç Ünlüyurt …….

Dr. Berrin Yanıkoğlu

…….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Dr. Gürdal Ertek for his encouragement, guidance, financial support and help for learning Java.

I would like to thank graduate committee members of my thesis, Dr. Tonguç Ünlüyurt and Dr. Berrin Yanıkoğlu for their critical suggestions and continuous guidance throughout this study.

I would also express my sincere gratitude to Dr. Ersin Özuğurlu and Dr. Kemal Kılıç for their advice, time and insight.

It is obvious that without strong friendship, emotional support, encouragement, and entertainment my graduate study in Sabancı University would be much harder. Therefore, I would like to thank my entire friends and especially my office-mates, Bilge Küçük, Evren Burcu Kıvanç and Özkan Öztürk. I will always remember the enjoyable days spent with Bilge as flat mates. Finally, I want to thank Mehmet Kayhan who makes our life cheerful and amusing while writing our thesis.

I am grateful to N.Mehmet Gökhan for spending time to arrange the order data with Excel. I am also grateful to all the graduate students and faculty members for providing a peaceful environment in the department.

I wish to thank my entire extended family for providing a loving environment for me. With their guidance, love and encouragement, I have overcome many formidable challenges throughout my life.

ABSTRACT

This thesis consists of two major parts. The first part is the presentation of two dynamic algorithms and the second part is the performance analysis of these algorithms using a real order data.

In this study, two algorithms, Slot-based Dynamic Algorithm and Order-based Dynamic Algorithm are developed for dynamic forward area allocation. Both algorithms are based on the idea of reassignment of most profitable items to the forward pick area. For Slot-based Algorithm, the profitable item is selected whenever slot becomes empty. This decision criterion is changed in Order-based Algorithm and becomes the end of definite order cycle. This thesis attempts to determine savings gained by implementing two algorithms in various warehouse settings. Instead of savings, number of forward picks, replenishments between reserve and forward areas and finally number of stock outs are analyzed as performance measures.

Earlier research focused mainly on the relocation of items only in the forward area according to the changing item popularity. On the other hand, our study addresses the issue of reassignment of items from reserve to forward slots in a volatile market environment. Additionally, the forward area is assumed to be a physical space consisting of slots, which makes our proposed algorithms convenient for real life applications.

Our study provides interesting insights in order to increase warehouse efficiency. We observe the advantages and disadvantages of both algorithms in different warehouse settings. We believe that these insights help managers to select the robust methods for creating forward area allocations.

ÖZET

Bu tez başlıca iki önemli kısımdan oluşmaktadır. Birinci kısımda iki dinamik algoritma, ikinci kısımda ise dinamik algoritmaların gerçek sipariş bilgilerinin kullanılması ile elde edilen performanslarının analizleri sunulmaktadır.

Bu çalışmada, depo ön alanının dinamik ataması için Bölme-odaklı ve Sipariş-odaklı olmak üzere iki algoritma geliştirilmiştir. İki algoritma da en fazla kazanç getirecek sipariş mallarının ön depo alanına atanması fikrine dayanır. Bölme odaklı algoritmada herhangi bir bölme boşaldığında, en fazla kazancı getirebilecek olan mal boşalan bölmeye atanır. Bu karar kriteri Sipariş-odaklı algoritmada değişir ve daha önceden tanımlanan sipariş devrinin sonu olarak belirlenir. Bu tez çalışmasında, iki algoritma farklı depo tasarımları için meydana getirdikleri kazançlar bakımından incelenmiştir. Kazanç yanında ön depo alanından yapılan sipariş mallarını toplama sayısı, arka depo alanından ön depo alanına yapılan mal ikmal sayısı ve talep edilen sipariş mallarının ön depo alanında yeteri kadar bulundurulamaması durumları performans kriterleri olarak kullanılabilir.

Geçmiş çalışmalar genellikle ön depo alanındaki malların yerleşiminde değişen müşteri taleplerine göre meydana gelen dinamik değişimleri ele alır. Bizim çalışmamız ise malların arka depo alanından ön depo alanına değişen taleplere göre atanması ile ilgilenir. Ek olarak, ön depo alanının fiziksel bölmelerden oluştuğu düşünülürse, bu yaklaşım ile geliştirdiğimiz algoritmaların gerçek yaşamda uygulanabilirliği de kabul edilebilir.

Çalışmamız depo kullanım etkinliğinin artırılması konusunda ilginç öngörüler sağlamaktadır. Her iki algoritmanın da avantaj ve dezavantajları farklı depo tasarımlar için gözlenmiştir. Bu öngörülerin yöneticilere doğru ön depo atama metotlarını seçebilmeleri için yardımcı olacağına inanıyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This research is concerned with dynamic configuration of forward slots. The answers of the following fundamental questions are investigated:

1. Which items are assigned to the forward area?
2. How much space is allocated to each?

These questions create forward reserve problem (FRP) in the warehouse literature. Hackman and Rosenblatt (1990) presented the first heuristic for the problem. Many researchers have contributed to this area. Generally, instead of physical storage spaces, the continuous forward area is used in the previous studies. Nothing is static in life; customer taste and demands change continually. Other economic factors such as marketing pressures for more diversified products and shorter product life cycles result in additional importance to warehouse management and policies. It is obvious that, all these changes require a dynamic warehouse management. Sadiq et al. (1995) and Jaikumar et al. (1990) analyze the broader area of relocation of items in dynamic warehouse systems. They focus on the location of items in the forward area instead of item types and item volumes.

We focus on the subject of reassignment of items to forward slots in dynamic and volatile market environment

Chapter 2 includes a comprehensive related literature review, warehouse terminology and explanation of basic issues.

Chapter 3 explains the order picking activity, forward and reserve areas and major trade offs of the forward pick area.

Chapter 4 describes forward reserve problem, conceptual framework and Hackman and Rosenblatt's heuristic. The detailed explanation of the fluid model is given in this chapter.

Chapter 5 presents our two dynamic algorithms: Slot-based and Order-based Dynamic Algorithms. These algorithms are generated in order to investigate possible forward area configurations for various warehouse settings.

Chapter 6 includes the experimental design. In this chapter, order data from a company is analyzed and different warehouse parameters are created for screening the effectiveness of proposed algorithms.

Chapter 7 reports the computational study of two algorithms for various warehouse designs. Both algorithms are observed individually and then their performances are compared. Important insights are obtained from the experiments. One of the most important of these insights is that the advantage of Slot-based Algorithm decreases with increasing number of slot. Order-based Algorithm with large order cycles generally overwhelm Slot-based Algorithm approximately for all experiments in terms of average saving.

Chapter 8 presents conclusions and outlines future works.

2. LITERATURE SURVEY

Warehousing is a significant cost component in supply chain activities, and deserves wide attention of researchers. Figure 2.1 depicts the cost components in supply chain operations.

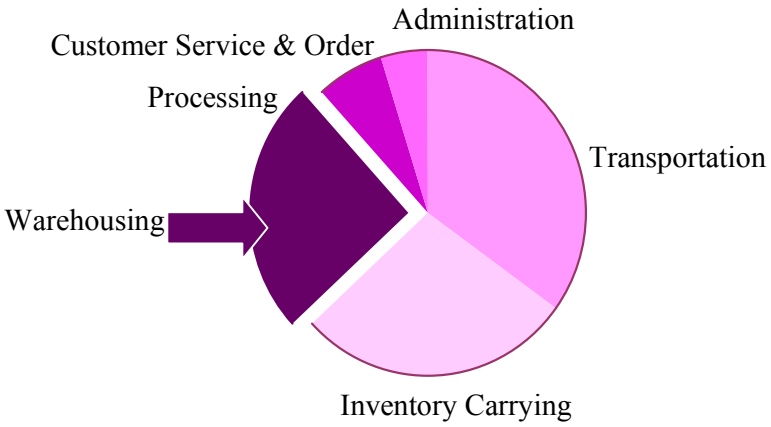


Figure 2.1. Costs in supply chain *(Frazelle, 2002)*

In today's competitive market, wider variety of products, short life cycles and importance of customer satisfaction force warehouse managers to improve planning and control of warehousing systems. Planning of warehousing systems includes policies concerning the assignment of products to storage locations. Control of warehousing problems involves the sequencing, scheduling and routing strategies. While planning algorithms considering an existing situation are based on historical data, control algorithms are based on actual data and concern to find solutions with a high-quality performance. Control of warehousing systems is outside the scope of this thesis.

Two step planning procedure is given as:

1. Distribution of products among warehousing systems

2. Assignment of products to storage locations

The relevant literature survey can be gathered under two main sections: Distribution of products in a typical warehouse and assignment of products to storage locations.

2.1. Distribution of Products in a Typical Warehouse

Most large warehouses use separate areas for order picking (forward area) and for bulk storage (reserve area). Whenever a product is depleted in the forward area, it is replenished from the reserve area. Various papers describe the development of assignment algorithms and policies for the forward area.

Hackman et al. (1990) formulate the Forward-Reserve Problem (FRP) and present a knapsack-based heuristic. In the authors' model, order picking from the reserve area is allowed and consequently the question of which items are assigned to the forward area and in what quantities arises. The objective of the FRP model is to maximize the net benefit, the difference between the saving from forward picking and cost of replenishment to forward area. Optimal item quantities are derived as a function of the available storage space. They present a knapsack-based heuristic that assigns these optimal quantities to the forward area according to the decreasing economic assignment quantity (EAQ) while the available forward space is filled.

Frazelle et al. (1994) examine an assignment-allocation (AA) sub problem to determine which items should be assigned to the forward area, and in what quantity. A formal definition of the forward-reserve problem (FRP) for a cart picking system is provided in this study. The costs of order picking and replenishment are related with the size of the forward area. Congestion constraint as well as volume capacity constraint are added to the model. The redundancy of the congestion constraint is proved. The authors demonstrate that the procedure in Hackman and Rosenblatt's (1990) research gives the optimal solution to the continuous relaxation of the problem. In their extensive study, the authors present a case study where 20% saving on labor cost is obtained by

diminishing the forward area to 32% of its original size.

Only little research has been done in the field of multiple-forward area assignment-allocation problem. Hackman et al. (1990) develop a mathematical programming procedure that solves a generic problem of allocating limited resources among several competing activities. In this study, the simple algorithm generates a near-optimal solution, when each allocation is the small fraction of the resource capacity. The prices associated with each resource are derived. These prices are calculated by nonsmooth optimization.

Van den Berg et al. (1995) present an integer-programming model for the forward-reserve problem that maximizes the expected number of picks from the forward pick area. A greedy heuristic is presented that attempts to improve the solution of the model. The authors also introduce an alternative model for minimizing total amount of work involved in order picking. Throughout the paper, it is assumed that items are replenished and stored in unit loads. Prior to the picking period, the entire forward area is replenished and whenever an item is depleted during the picking period, replenishment activity is performed.

2.2. Assignment of Products to Storage Locations

Three storage location assignment policies are introduced: *Randomized storage, class-based storage and dedicated storage.* Under the randomized storage policy, the products are stored anywhere in the storage area. The class-based storage policy distributes the products based on their activity level, size, environmental requirements etc. Under the dedicated storage policy each location may only be used for a specific product. Randomized and class-based storage are also known as *shared storage*, meaning that different products are allowed to be stored in the same location.

Sharp et al. (1998) discuss the traditional product storage assignment concepts, including the cube-per-order index and results for the forward-versus-reserve allocation.

The results are discussed in the context of a typical large distribution center that holds many products and where more than one person may select items for an order. They comment that textbook formulas giving the space savings as a result of using shared (instead of dedicated) storage are too optimistic due to their ignoring product demand variability and correlations among product demands. They examine also the effects of these factors, as well as the frequency of product re-assignment. They state that the related issue of storage compartment size affects the forward-versus-reserve allocation and that the traditional approach is to treat the storage compartment size as being infinitely variable. The authors interpret that this approach for pallet and carton flow rack storage systems are not true. They give case studied where there is a consideration of matching the compartment sizes with the product re-order quantities; the results include major savings in labor. They propose a cluster analysis procedure used to obtain various correlation measures, to create nested clusters, and to give the performance analysis. The method yields dramatic improvements compared to traditional activity-based storage in some applications.

More recent studies have focused on dynamic relocation, such as Jaikumar et al. (1990) and Sadiq et al. (1996).

Jaikumar et al. (1990) analyze the optimal relocation of pallets with a high expectancy of retrieval within each storage rack of an automated warehouse to meet the changing demand patterns. They develop certain conditions that an optimal relocation policy satisfy and design a very efficient optimal relocation algorithm. The relocation of pallets with high expected demand closer to the input / output point of each rack reduce the expected travel time.

Sadiq et al. (1996) introduce Dynamic Stock Location Assignment Algorithm (SLAA), an improvement algorithm for dynamic warehouse planning. The problem of stock location assignments in an in-the-aisle order picking system to minimize the order picking time is addressed. SLAA utilizes the future product mix, product structure and demand forecasts when assigning an item to order picking systems. Generally, the algorithm can be divided into two phases: the global assignment phase and the local

assignment phase. The main purpose of the global assignment phase is to perform capacity analysis. In this phase, future demands, slot size constraints and history usage are considered for the decision making process about which items should be in the system. After the capacity analysis is performed in the global assignment phase, in the local assignment phase, a two-phase (hybrid) clustering technique, HYCLUS is examined. HYCLUS considers the common properties of items on orders and utilizes a hierarchical clustering to provide the starting number of clusters for the non-hierarchical phase, which begins with the hierarchical solution and converges to a local optimum.

Dynamic slot allocation, which is developed in his thesis, is considered during the distribution of items in the warehouse.

Before analyzing the common order picking activities, some important terms are defined:

Stock-Keeping-Unit (SKU): The unit of measure in which an item is stocked.
Item: The smallest unit of a product.
Order: A document requesting specific SKUs in specific quantities.
Pallet: "A set of cartons or totes of identical product arranged in a cubical pattern and usually supported by a base that made of wood or plastic'' (*Sharp, 2000*).
Slot: A small division that is fully allocated to a single item.
Storage: The physical shape of items while staying in the warehouse.
Replenishment: When the inventory of an item assigned to the forward area reaches its critical level, an amount of stock corresponding to the size of the location in the forward area is retrieved from the reserve area, transported and stocked in the forward area. This activity is called replenishment.

3. ORDER PICKING

Order picking is defined as the retrieval of the appropriate amounts of products from a pick (or storage) area to fulfill customer orders. Orders are usually represented as a list of SKU's. Also order picking can be defined as the process of identifying, selecting, retrieving and accumulating the items on customer orders. 55% of all operating costs in a typical warehouse could be attributed to order picking (Frazelle, 2002). The distribution of major costs in a typical warehouse is represented in Figure 3.1.

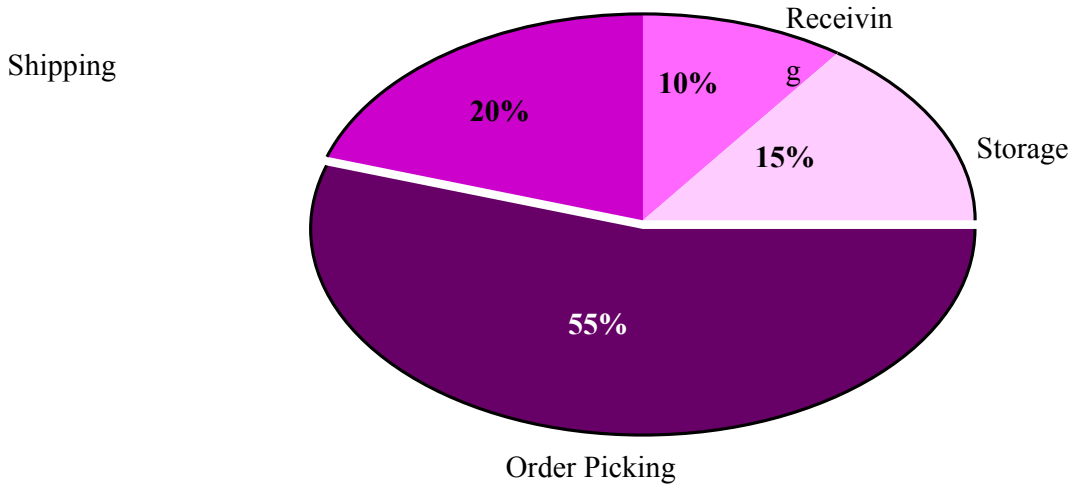


Figure 3.1. Operational costs in a warehouse (*Frazelle, 2002*)

Figure 3.1 demonstrates that the order picking is the most costly and critical activity in a warehouse.

The order picker generally deals with the following activities in a picker-to-stock system:

- *Traveling* to, from, and between pick locations
- *Extracting* items from storage locations
- *Reaching* and *Bending* to access pick locations
- *Documenting* picking transactions
- *Sorting* items into orders
- *Packing* items
- *Searching* for pick locations

Among these activities, the most time consuming ones are traveling and searching. Figure 3.2 illustrates the operational order picking times

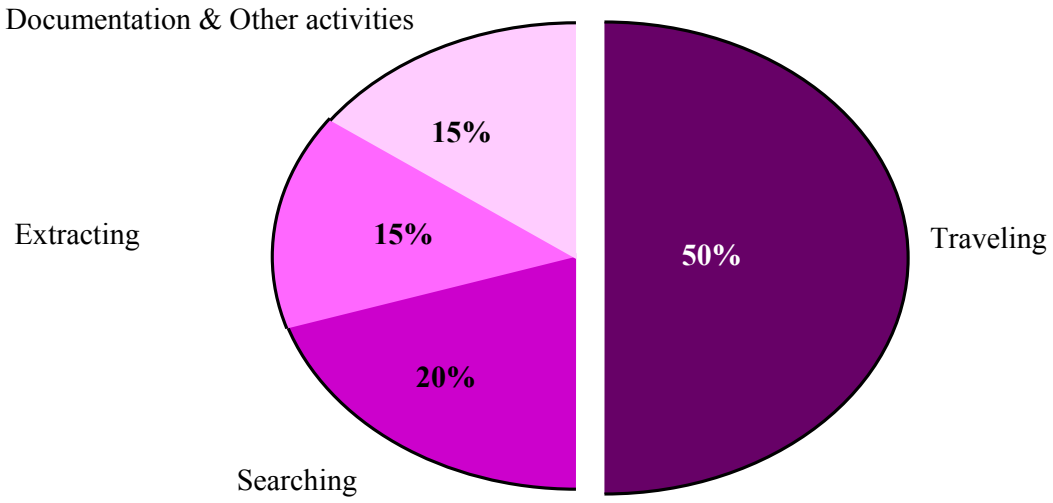


Figure 3.2. Typical distribution of an order picker's working time (*Frazelle, 2002*)

Establishing separate forward and reserve picking areas generally improves the traveling and searching times of order pickers in a typical warehouse. Forward and reserve picking areas will be examined in detail.

3.1. Forward-Reserve Pick Areas

An efficient approach to reduce the amount of time associated with order picking is to divide the warehouse into a *forward* area and a *reserve* area. Forward area is usually used to store items with the high forecasted demand in relatively small amounts

for minimizing the order picking time and for increasing responsiveness to customer demand. Forward area is sometimes called *fast-pick* or *primary pick area*. Figure 3.3 shows a typical forward pick area.

Figure 3.3. A typical forward pick area in a warehouse

Reserve area is used to replenish the forward area and to pick the items that are not assigned to the forward area. Reserve area generally holds the bulk storage and is sometimes called *secondary pick area*. Figure 3.4 represents a typical reserve area in a warehouse.

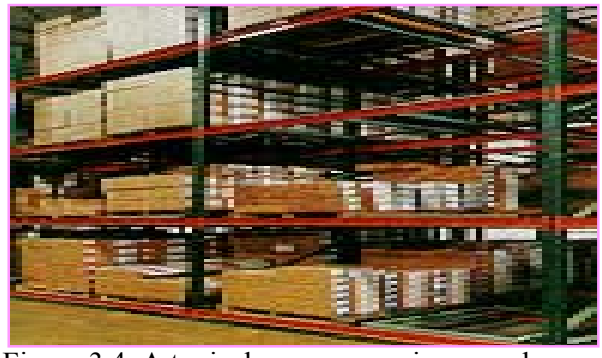


Figure 3.4. A typical reserve area in a warehouse

In some warehouses, the reserve area is separated into two areas: one for order picking and one for replenishment. The forward and reserve areas may be distinct areas

or they may be located in the same rack, where the lower levels represent the forward area and higher levels represent reserve area. Figure 3.5 illustrates the basic structure of the interactions in a forward/reserve system.

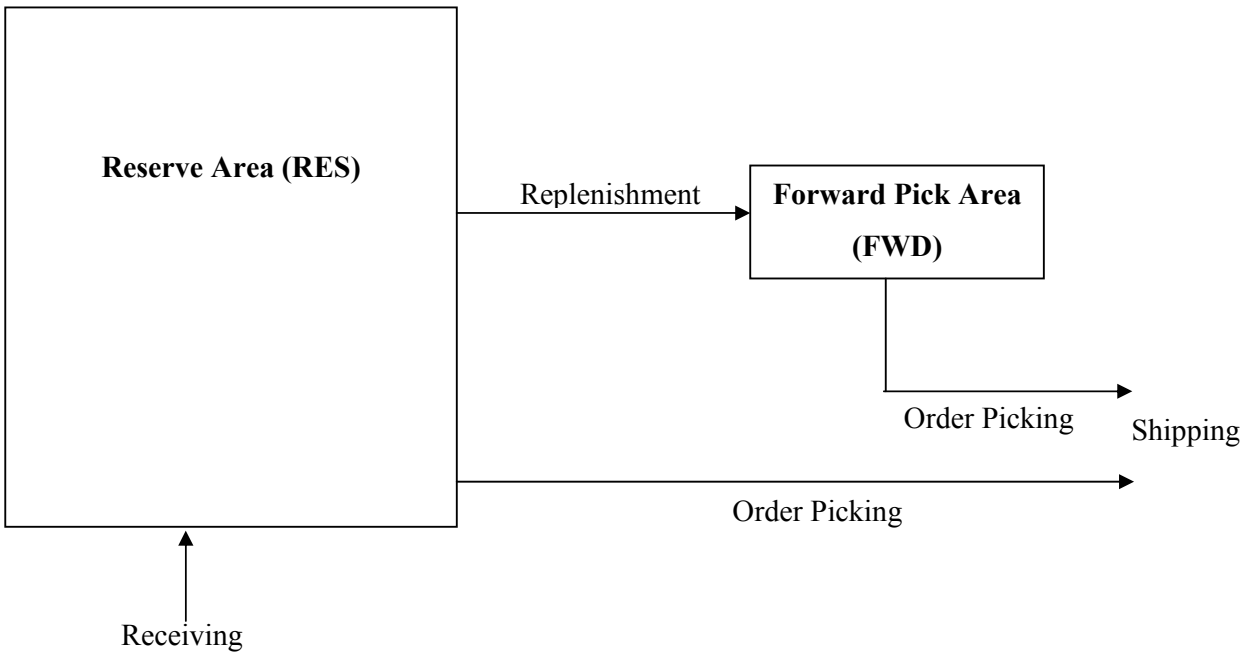


Figure 3.5. Interactions between forward and reserve areas

The arrows represent a material flow pattern for items stored in the forward pick area and reserve area. Incoming items are received and then stored in pallets in reserve storage. As orders are received for items, the items are picked from the forward pick area and shipped. When the inventory levels of items in the forward pick area drop to a critical level (threshold), they are replenished from reserve area. The items that are not found in the forward pick area are picked directly from the reserve area.

3.1.1. Major Tradeoffs of a Forward Pick Area

There are tradeoffs and cost considerations involved establishing a forward pick area. A separate forward pick area generally increases the pick density in a typical warehouse due to following reasons:

- The forward pick area is a relatively small area
- The traveling and searching efforts of order pickers are reduced

In order to decrease traveling and searching costs of order picking activity in the forward area, it is better to keep it as small as possible. However, the small forward area leads to more frequent replenishment trips between forward and reserve areas. These trips mean more staffing requirement and replenishment cost.

A small pick area decreases picking costs but increases replenishment costs. The design of the forward pick area is a strategic decision. The basic issues in the design of a forward area are:

- Determining the size of the forward pick area
- Determining the set of items to be stored in the forward pick area
- Determining the amount of each item to be stored in the forward pick area
- Determining the storage technologies to be used

The following cost components are relevant to the establishing separate forward reserve pick areas:

- Capital cost of equipment
- Labor costs for order picking activity

Finding the optimal critical space allocation for each item in the forward area that minimizes the order picking, replenishment and storage equipment costs is known as the forward / reserve problem (FRP) which will be examined fully in the next chapter.

4. FORWARD / RESERVE PROBLEM

The *forward-reserve problem* (FRP) is the problem deciding which items should be stored in the forward area and in what quantities. FRP is an important problem while assigning an item to the forward area means reduced order picking costs but increased replenishment costs. The tradeoff between replenishment and order picking costs makes FRP an important problem studied by several researchers. Hackman et al**.** (1990) are the first to present a model for FRP problem that considers both assignment and allocation. They describe a heuristic that attempts to maximize the total net benefit The *economic assignment quotient* (EAQ) is also first used in Hackman and Rosenblatt's paper (1990). EAQ is used to design a simple algorithm that solves the FRP to near-optimality.

Generally, the algorithms that are developed for forward / reserve problem use some key statistics for each item such as flow and picks. Picks are measured by pick-lines per period and flow is measured in cubic-feet per period. These statistics can be forecasts or historical data.

The following notations are used throughout this chapter except section 4.1 where the original notations are used:

C_r : Cost of each replenishment trip

S : Saving when a pick is made from the forward pick area

V : Total volume of the forward pick area (feet cube)

N : Number of items in the warehouse system

For each item i

D_i : Demand in units / period

F_i : (Flow) demand in feet cube/ period

P_i : Number of picks/ period

$H_i(U_i)$: Saving from allocating volume U_i to item *i* in the forward pick area

w_i : Volume of item *i* (feet cube)

U_i : Volume assigned to item *i* (feet cube)

Flow of item i, F_i, is described as:

$$F_i = D_i w_i \qquad (1)$$

Where flow F_i has been expressed as the product of demand D_i and volume w_i.

In the next section Hackman and Rosenblatt's FRP model is introduced.

4.1. Hackman and Rosenblatt's FRP Model

Hackman and Rosenblatt formulate the FRP and provide a heuristic to the assignment allocation sub problem. They deal with allocating items to an automated storage and retrieval system (AS/ RS). While the capacity of the AS/RS is insufficient to store all the items, the questions of which items to assign to the AS/RS and what quantities are examined. The following simplifying assumptions are made throughout the paper (Hackman, 1990).

1. "The demand process and all cost data are assumed stationary in continuous time over an infinite horizon, as in the standard EOQ model.
2. Reserve areas (secondary locations) have infinite capacity.
3. The on-hand inventory in the forward area is always sufficient to accommodate any internal replenishment.
4. The material handling cost to complete a customer request or an internal replenishment is its size.
5. The physical dimensions of items assigned to the AS/RS are very small."

Let

"e_i = Savings per request for item *i* if stored in the AS/RS

c_i = Cost per replenishment for item *i*

R_i = The number of requests per unit time for item *i*

D_i = The demand per unit time for item *i* converted into units of volume

z_i = Continuous decision variable determining the space in the AS/RS allocated to item *i*

x_i = Binary (0 or 1) decision variable determining if item *i* is assigned to the AS/RS

$f_i(z_i)$ = Profit per unit time for item *i*

N = Number of items in the warehouse

V = The volume of the AS/RS"

All parameters are assumed positive. The net benefit of storing item *i* in the forward pick area can be expressed as:

Net Benefit = Total Pick Saving – Total Replenishment Costs

The net benefit of storing z_i cubic feet of item *i* in the forward area is formalized as follows:

$$f_i(z_i) = \begin{cases} e_i R_i - \dfrac{c_i D_i}{z_i} & \text{if } z_i > 0 \\ 0 & \text{if } z_i = 0 \end{cases}$$

The allocation model is formulated as:

$$\max \sum_{i=1}^{N} f_i(z_i)$$

s.t.

$$\sum_{i=1}^{N} z_i \le V$$

This model is used to make forward-reserve decision. The following three remarks are examined in Hackman and Rosenblatt's paper (Hackman, 1990).

1. Each item *i* has a minimum threshold volume:

$$\frac{c_i D_i}{e_i R_i} \qquad i=1,\ldots,N$$

If the volume of the item *i*, z_i is below this value, it does not pay to assign this item to the forward area. The costs of replenishment can outweigh any savings in pick from the forward area.

2. The priority of each item i is evaluated as:

$$\frac{R_i}{\sqrt{D_i}} \qquad i=1,\ldots,N$$

Where R_i is the number of requests per unit time for item *i* and D_i is the demand per unit time for item *i* converted into units of volume. This formula is called *economic assignment quotient* (EAQ). The items with the greatest EAQ are selected for the forward pick area.

3. Hackman and Rosenblatt introduce the square-root space allocation formula. The following volume is assigned to each selected item i

$$z_i = \frac{\sqrt{D_i}}{\sum_j \sqrt{D_i}} V$$

D_i is the demand per unit time for item *i* converted into units of volume and *V* is the total volume of the forward area. *j* represents the subset of items that are chosen to go into the forward area.

Hackman and Roseblatt's algorithm can be summarized as follows:

Step 1: Sort all items from largest EAQ to smallest. In the case of a tie, put the item with the largest D_i value.

Step 2: Determine the net benefit of assigning first item to the forward area, first two

items to the forward area and so on. Select the set that maximizes the total net benefit.
Step 3: In the selected set, start with the last item, having the smallest EAQ value, and check whether the item's assigned volume to the forward area exceeds the threshold volume or not. If it is below the threshold volume, the item is eliminated from the set.
Step 4: Obtain the last feasible set and calculate the total net benefit.

Hackman and Rosenblatt's research depends on the *fluid model*, which constitutes the basic structure of this thesis. In the next section the fluid model is examined in detail.

4.2. Fluid Model

In the fluid model each item is treated as an incompressible continuously divisible fluid; the fact that an item comes to the warehouse as a discrete space such as pallets, cases or individual units is completely ignored. In this simple method, the cubic feet of storage space that is allocated to each item is measured. Each item is assumed to be small enough to be replenished in case quantities. The fluid model represents the ideal situation for the warehouse system.

In the following sections cost of replenishment, allocating space in the forward area, storage heuristics, items that go into the forward pick area will be examined respectively.

4.2.1.Estimate the Cost of Replenishment

The cost of storing item i in the forward pick area is the replenishment cost of item i. The cost of replenishment from reserve to forward is based mostly on the number of replenishments required. The number of replenishments depend on the type of the storage unit; if the items are stored as pallets then each pallet requires separate handling. On the other hand if the items are stored in small containers, a fluid model can estimate the number of replenishments:

Let F_i be the flow of item i per period and U_i is the volume assigned to item i in the forward area. Then the number of replenishments per period is calculated as:

$$\frac{F_i}{U_i} \qquad (2)$$

If the cost of each replenishment is C_r then the cost per period of storing U_i cubic feet of item *i* is estimated as:

$$C_r \frac{F_i}{U_i} \qquad (3)$$

$\frac{F_i}{U_i}$ represents the number of replenishments per period

(3) holds two important assumptions:

1. Item i is replenished to the forward area only after the item i's in the forward area is completely consumed,
2. Cost of each replenishment is independent of the quantity replenished. This is valid for only small items.

4.2.2. Allocating Space in the Forward Pick Area

One of the most important questions of the FRP is how much space is allocated to each item. For the simplest case, assume that every item is represented in the forward pick area of volume V. Figure 4.1 illustrates this case (Bartholdi, 2000).

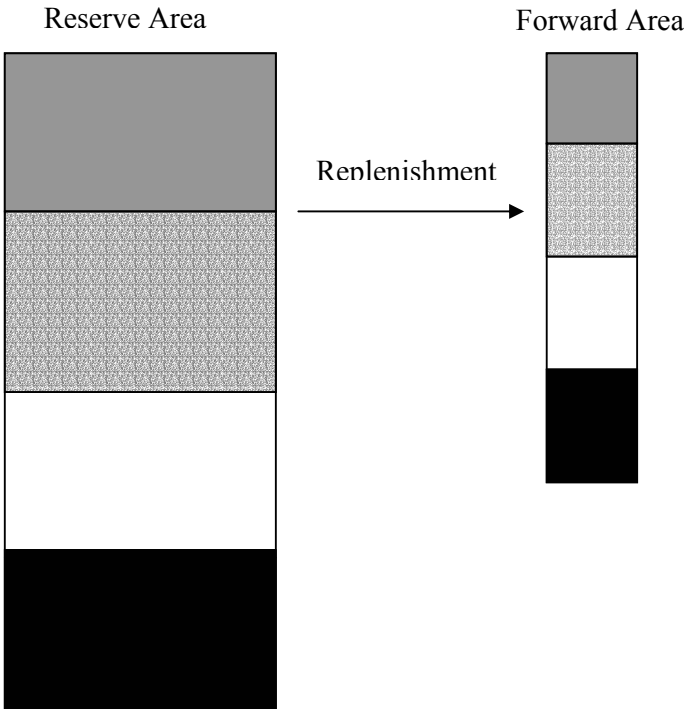


Figure 4.1. Case where every item is represented in the forward area (*Bartholdi, 2000*)

The main aim is to store the right amount of every item for minimizing the total cost of replenishment. The model can be constructed as follows: (*Bartholdi, 2000*)

$$\min \sum_{i}^{N} C_r \frac{F_i}{U_i}$$

s.t.

$$\sum_{i}^{N} U_i \le V$$

$$U_i \ge 0$$

By eliminating the volume constraint, and bringing it into the objective function with a Lagrangean multiplier takes the following form: (*Bartholdi, 2000*)

$$\min \sum_{i}^{N} \frac{F_i}{U_i} - \lambda(\sum U_i - V)$$

$$U_i \ge 0$$

Lagrangean variable λ can be interpreted as the "rent" charged to each item for

storage space.

For each separate U_i:

$$U_i = \sqrt{\frac{F_i}{\lambda}}$$

After setting $\sum_i U_i = V$, it is found that $\lambda = \left(\sum_i \sqrt{\frac{F_i}{V}} \right)^2$. If this expression is substituted into that for U_i, the new expression gives (4).

In order to minimize the total replenishment cost over all items, j=1,...,N, each item should be stored in the amount of:

$$U_i^* = \left(\frac{\sqrt{F_i}}{\sum_{j=1}^{N} \sqrt{F_i}} \right) V \qquad (4)$$

Where U_i^* is the optimal amount of space that should be allocated to item i in the forward pick area. Unfortunately, this amount may be inapplicable in practice. For example, in flow rack at least an entire lane is allocated to each item.

The optimal number of replenishments per period is denoted as:

$$\frac{F_i}{U_i^*} \qquad (5)$$

The optimal number of replenishments per period per cubic foot of item i, $\frac{F_i}{\left(U_i^*\right)^2}$, is rewritten by using (4) and found to be independent of item i. The optimal number of replenishments per period per cubic foot of item i:

$$\frac{\sum_{j=1}^{N} F_j}{V^2} \qquad (6)$$

Therefore each part of the forward storage area is replenished at the same rate.

4.2.3. Storage Heuristics

Some heuristics are used other than optimal volume allocation in (4), to determine how to allocate space among items. These heuristics are explained below:

1. Equal Space Allocation: The same space is allocated to every item in the forward area. If V cubic feet is available and there is N items, then

$$U_i = \frac{V}{N}$$

and item i is replenished $\frac{NF_i}{V}$ times a period

2. Equal Time Allocation: An amount of space sufficient to meet the demand over a specified period is allocated to every item. Equal Time Allocation is denoted as:

$$U_i = \left(\frac{F_i}{\sum_j F_j} \right) V$$

In Equal Time Allocation, item i is replenished $\frac{\sum_j F_j}{V}$ times a period

Equal Time Allocation performs no better than the Equal Space Allocation. If the number of replenishments per period is compared, both policies give the same number of replenishments:

$$\frac{NF_i}{V} = \frac{\sum_j F_j}{V}$$

In the warehousing industry, it is common to apply Equal Time Allocation after implementing ABC analysis. Optimal allocation policy, which is explained in section 4.2.2, is used in this thesis**.**

4.2.4. Items That Go into the Forward Pick Area

Some items, so called *slow moving*, are less popular than the others. It does not make sense to store such items in the forward pick area. If more popular items, so called *fast moving*, are stored in the forward pick area, replenishment activities from reserve storage to forward pick area are reduced. Slow moving items are picked directly from the reserve storage area, which is more expensive than picking from the forward pick area. Therefore the situation becomes like that of Figure 4.2:

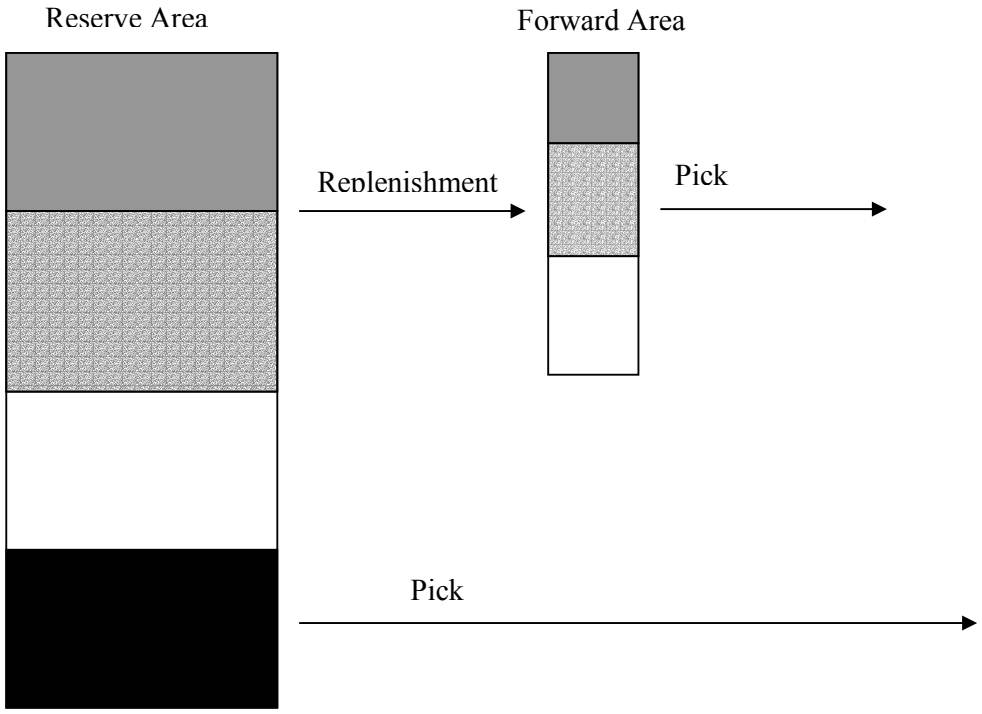


Figure 4.2. Some items represented in the forward pick area (*Bartholdi, 2000*)

To better answer the questions of which items should go into forward pick area and which ones stay in the reserve storage area, the net benefit of storing U_i cubic feet of item i in the forward area is examined as follows:

$$H_i(U_i) = \begin{cases} 0 & \text{If } U_i = 0 \\ SP_i - C_r \dfrac{F_i}{U_i} & \text{If } U_i > 0 \end{cases} \qquad (7)$$

The right subset of items and in the right amounts are tried to be stored in order to maximize the net benefit. In the model, it is assumed that the reserve area is sufficiently large. The saving model is expressed as follows:

$$\begin{aligned} Max \quad & \sum_{i=1}^{N} H_i(U_i) \\ \text{s.t.} \quad & \\ & \sum_{i=1}^{N} U_i \le V \\ & U_i \ge 0 \qquad \forall i \end{aligned} \tag{8}$$

It is aimed to maximize the saving function $H_i(U_i)$ for all the items in the warehouse. The total volume assigned to the items in the forward pick area cannot exceed the total volume of the forward area.

There is a minimum sensible amount of each item to store in the forward pick area:

$$\frac{C_r F_i}{SP_i} \tag{9}$$

This useful managerial expression is proved by solving $SP_i - C_r \frac{F_i}{U_i} = 0$ and the value of U_i that results in a net benefit of 0 is observed. Figure 4.3 illustrates the minimum sensible amount of each item in the forward pick area.

If too little of item *i* is put in the forward pick area, the forward area has to be replenished so frequently that the total replenishment costs exceed the saving obtained by picking directly from the forward pick area.

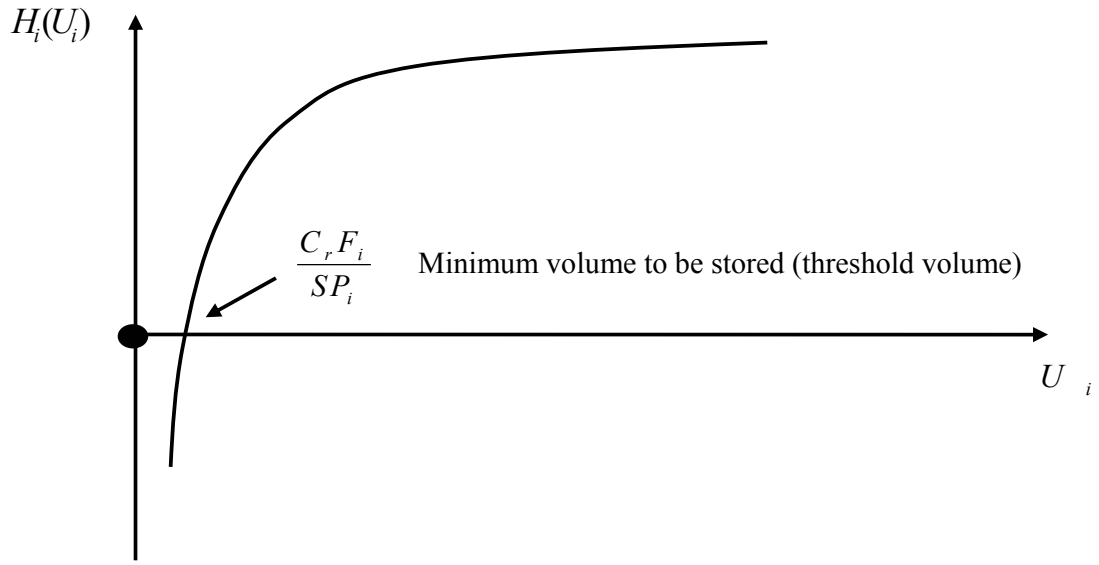


Figure 4.3. The net benefit function (*Bartholdi, 2000*)

The saving model, (8) is first introduced by Hackman and Rosenblatt (*Hackman, 1990*). Their theory and heuristic was explained in section 4.1. They present a near optimal solution to the model: The items that have the strongest claim to the forward pick area are those with the greatest viscosities:

$$\frac{P_i}{\sqrt{F_i}} \qquad (10)$$

The viscosity, sometimes called Cube Per-Order-Index (CPOI), is an important expression helps choosing the items to put in the forward pick area. It represents the activity level of the item. Also, it measures effort (labor) required to move a given flow from the warehouse.

Hackman and Rosenblatt's heuristic that finds the appropriate item subset for the forward pick area allocation is explained in section 4.1. While the heuristic allows the use of continuous quantities, the physical storage area is completely ignored. Dynamic slot allocation, which converts the continuous storage space approach to discrete storage space, is built in this thesis.

5. DYNAMIC ALGORITHMS

In the global market, customer demands change rapidly and show great variety. Meanwhile, shorter life cycles are observed for items. In addition there are seasonality and different promotional programs. All these factors create the need for a dynamic approach to warehouse management. Due to changing item demands; different items may be transferred from reserve to forward area in each replenishment period. Sadiq et al. (1995) and Jaikumar et al. (1990) analyze the broader problem of item location assignments in dynamic warehouse systems. They focus on the location of items in the forward area instead of item types and item volumes.

In this thesis, two algorithms, Slot-based Dynamic Algorithm and Order-based Dynamic Algorithm, are developed in order to generate different forward area configurations. The item types and assigned volumes in the forward area are observed in each algorithm. The main question is which items are assigned to forward area and how many slots are allocated to these items.

The logic behind two developed algorithms requires a dynamic approach where items that yield small labor savings are removed from the forward area and exchanged by more "profitable'' items. We investigate a set of future orders in order to select the most profitable item. It should be noted that, the most profitable items is change every time a decision is made in the dynamic market conditions.

In the fluid model, the storage area is assumed to be continuous which is not observed in real life applications. Therefore, the geometry of the storage area is discretized in our algorithms. Items are assumed to be stored in slots in the forward pick area. The results of the fluid model are rounded to the closest allowable amount.

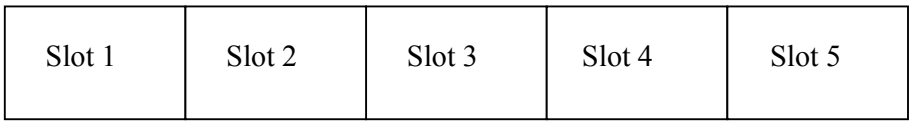


Figure 5.1. Discrete forward area design

The design of the typical forward area consisting of slots is illustrated in Figure 5.1. The slots in the forward area have the same dimensions and also same volume. The items in the forward area are assumed to be stored in these physical areas.

Developed algorithms use distinct decision criterions for designating the reassignment time. In the Slot-based Algorithm, when a slot or slots become empty, reassignment of items begins. More popular item is replenished from reserve area to empty slots. In other words; emptying of a slot triggers a decision in the Slot-based Algorithm. On the other hand, "order cycle" is the decision criterion for reassignment time in the Order-based Algorithm. Order cycle is the definite order value: For example if 300 orders were defined as the order cycle, a replenishment is made to the empty slots in the forward area only after 300 orders.

The following assumptions are used in the algorithms:

- A fully operational warehouse with a forward slot area and a reserve area is considered,A slot belongs to only one item type,Reserve area has infinite capacity,
- The item volumes are small enough so that assuming continuity is possible,
- Replenishments can be made continuously over time, and take negligible amount of time,
- If an item is picked from the warehouse the pick value is increased by one,
- Each slot has the same dimensions,
- The demanded items are first picked from the forward area,
- There is no congestion issue to consider.

5.1. Slot -based and Order-based Dynamic Algorithms

In this section, two algorithms are presented. Code validation is presented in Appendix A. The pseudo codes are given in Appendix B, C and D. Both algorithms consist of two sub-algorithms:

1. Initial Allocation
2. Dynamic Allocation

The Initial Allocation Sub-Algorithm has the same structure in both algorithms.

5.1.1. Initial Allocation Sub-Algorithm

This algorithm generates initial forward slot allocation. The initial item types and their assigned volumes are evaluated in this algorithm.

1. (Sort)

1.1. Sort all items from largest viscosity to smallest

2. (Select the item set)

2.1. Determine the total saving of assigning first item to the forward area, first two items to the forward area and so on.
2.2. Select the set that maximizes the total saving

3. (Check)

3.1. Start with the last item having the smallest viscosity in the selected set
3.2. Calculate the item's assigned volume in the forward area
3.3. Check whether assigned volume below the threshold volume or not

4. (Update)

4.1. If the assigned volume below the threshold volume, remove that item from the set. Return to Step 3 until all items in the set is analyzed
4.2. Obtain the last feasible set

5. (Discrete Allocation)

5.1. The assigned volume in the forward area is discretized for all items in the feasible set
5.2. Return the initial allocation data to Dynamic Sub-Algorithms

5.1.2. Slot-based Dynamic Sub-Algorithm

Slot-based Dynamic Sub-Algorithm uses the initial allocation obtained from Initial Allocation Sub-Algorithm. In this algorithm, whenever slots become empty we look ahead next *k* orders and investigate the items in these orders in terms of total saving. Then, the selected item is replenished to the empty slots.

1. ***(Generate a forward area)***

   1.1 Generate a forward pick area using the initial allocation data before picking the demanded items
2. ***(Process orders)***

   2.1. Start processing first order
   2.2. Define the demanded items
3. ***(Check)***

   3.1 Check whether demanded item exists in forward slots or not
   3.2. If it does not exist, return to Step 2 and process the next order
4. ***(Pick and Update)***

   4.1. Pick the demanded item from the forward slots.
   4.2. Update the remaining item volume in the forward area
5. ***(Check the empty slots)***

   5.1. Check whether slots become empty or not in the forward area after picking
6. ***(Make a decision)***

   6.1. Generate a set consisting of next *k* orders items
   6.2. Calculate total saving of assigning each item to the empty slot
   6.3. If all the savings are negative then empty slots remain empty
   6.4. Select the item having the maximum total saving
7. ***(Replenish and Update)***

   7.1. Replenish the selected item to the forward empty slot
   7.2. Update item data
   7.3. Return to Step 2 and process the next order

5.1.3. Order-based Dynamic Sub-Algorithm

Order-based Dynamic Sub-Algorithm uses the initial allocation obtained from Initial Allocation Sub-Algorithm. In this algorithm, whenever an order cycle finishes, we look ahead next *k* orders and investigate the items in these orders in terms of total saving. After every order cycle a decision is made about filling the empty slots with selected item.

1. (Generate a forward area)

1.1. Generate a forward pick area using the initial allocation data before picking the demanded items

2. (Process orders and check the index)

2.1. Start processing first order
2.2.Increase order index by one
2.3. Define the demanded items

3. (Check the order index)

3.1. If the order index equals to definite order cycle, Go to Step 6

4. (Check item)

4.1. Check whether demanded item exists in forward slots or not
4.2. If it does not exist, Return to Step 2 and process the next order

5. (Pick and Update)

5.1. Pick the demanded item from the forward slots.
5.2. Update the remaining item volume in the forward area
5.3. Calculate the total empty slot
5.4. Return to Step 2 and process the next order

6. (Make a decision)

6.1. Generate a set consisting of next *k* orders items
6.2. Calculate total saving of assigning each item to the empty slot
6.3. If all the savings are negative then empty slots remain empty
6.4. Select the item having the maximum total saving

7. (Replenish and Update)

7.1. Replenish the selected item to the forward empty slot
7.2. Initialize the order index
7.3. Update item data
7.4. Return to Step 2 and process the next order

6. EXPERIMENTAL DESIGN

6.1. Analysis of Test Data

In this section, the data set that is used for testing the effectiveness of the algorithms and the programming environment is described and analyzed.

S. P. Richards Company's (SPR) 2002 sales database, which was downloaded from Dr. John Bartholdi's web site at Georgia Institute of Technology in February 2003, is used in this thesis (*http://www.isye.gatech.edu/people/faculy/John_Bartholdi*). SPR was established in 1848 in Atlanta, Georgia. The company has over 150 years experience in office products industry. Genuine Parts purchased SPR in 1975. SPR is a leading firm with the following strengths:

- 36 full stocking distribution centers
- 22,336 catalogued items in the inventory
- Over 30,000 total items in inventory

S. P. Richards has a wide range of product profiles. Major class descriptions are the following: (1) Albums and frames, (2) binders, portfolios, sheet protectors, (3) break room supplies, (4) business bags, cases and accessories, (5) business books, records and forms, (6) computer accessory and software, (7) desk accessories, (8) drafting and engineering supply, (9) filing supplies and accessories, (10) janitorial supplies, (11) mailing supplies, (12) meeting and presentation supplies, (13) notebooks and pads, (14) office equipment and machines, (15) office machine consumable supplies, (16) office furniture and accessories, (17) office storage, (18) paper, (19) school supplies, (20)

home/office supply, (21) tapes and adhesives, (22) writing instruments.

The items in the company warehouses are replenished and stored within various types: *Bag, bottle, box, case, carton, pallet, each, etc*. In the dynamic slot allocation, the fluid model, which allows the usage of small quantities, is determined. Therefore, the items stored and / or replenished as *each* are taken from the sales 2002 database. The items are stored in slots in the forward pick area.

6.1.1. Analysis of Items

There exist 17,877 different item types in the S. P. Richards Company dataset. Each item type is illustrated as the combination of letters of alphabet and digits, called *item code*. Some of the items are replenished and picked from forward pick area where others stay in the reserve area. First four element of item code represents the vendor. The items are grouped according to the first four elements of the item code and the number of item types is reduced to a moderate value. The complexity of long item codes is diminished in this way.

In the reduced dataset, there are 29,737 orders and 734 different item types. Table 6.1 illustrates the first nine order in the dataset as an example:

Order / Item	1	2	3	4	5	6	7	8	9
ACM1	1								
HEWC	1								
CCP9		1							
BORE			1						
LIO4				1					
AKM3					40				
ITYM						2			
AVE4									1

Table 6.1. First nine orders of the selected SPR dataset

In the dataset, average number of stock keeping units (SKU) requested per order is 1.42. The related histograms and distributions of the number of SKU requests per order is given in Appendices E and F. Chi Square Test is also applied to the selected dataset in order to fit statistical distributions.

Order size is determined by cubic volume and/or number of different stock keeping units on the order. In literature, it is stated that a *small order* contains fewer than 10 SKU and a *large order* has 10 or more SKU requests (*Sharp, 2000*). Cubic volume approach declares that a small order is also limited to 3.5 feet cube if the volume per item is less than 0.35 feet cube in a small order. A large order has a volume greater than 3.5 feet cube (*Sharp, 2000*). The number of SKU requests and cubic volume demarcations are used in Section 6.1.3

SKU request per order changes between 1 and 36. It is observed that 29606 orders consist of less than 10 SKUs and 131 orders consist of 10 and more SKU requests.

6.1.2 Imaginary D Zone

SPR increases picking efficiency by dividing the warehouse into C and D zones. C zone is used for bulk storage and functions like a reserve area. On the other hand D zone can be thought as a forward pick area. Small items are allocated to the D zone.

In the original D zone layout, there are 3,286 slots, each slot dimension is 3.5 feet and one slot volume is 42.875 feet cube. The original forward are volume is the product of total number of slot and one slot volume:

$$\text{Original D zone volume} \cong 140{,}887 \text{ feet cube}$$

734 item types are used instead of 17877 item types. Therefore, the volume of imaginary D zone created for 734 item types is less than the original D zone volume. The following volume approach is used:

- The imaginary D zone volume is linear proportional to original D zone volume. If 3,286 slots are available for 17,877 item types, then 135 slots should be available for 734 item types. Imaginary D zone volume is approximately 5,790 feet cube.

Volume of the imaginary forward pick area = 135 slots*42.875 feet cube
=5,788.125 feet cube,
≅ 5,790 feet cube.

6.1.3. Experimental Data Sets

The effectiveness of the algorithms is evaluated by setting different values for parameters that characterize the warehouse:

- Number of initial orders that creates the initial allocation,
- Look ahead order number,
- Unit volume of the items (feet cube),
- Saving when a pick is made from the forward pick area,
- Cost of each replenishment trip,
- Number of slots in the forward area,
- Volume of each slot in the forward area (feet cube),
- Order cycle

In the experiments, the parameters are changed between some limits and various experimental datasets are used. These constraints are explained below:

- In the experiments the following warehouse design is considered:
  - The total forward area volume is 5,790 feet cube and is the same in all experiments.
- The product of the number of slots and slot volume must give total forward area volume.
- SPR selected dataset consists of mostly small orders, as explained in

section 6.1.1. Therefore, the item volumes are less than 0.35 feet cube. The maximum item volume is 0.349 and minimum item volume is 0.07 feet cube.

- The sum of initial order number and look ahead order number must smaller than the whole order number.
- The saving when a pick is made from the forward area is smaller than the cost of each replenishment trip. (7) gives the related saving function.

For different experimental datasets, various statistics are collected and analyzed. These statistics are used for evaluating the effectiveness of the algorithms:

- Number of forward picks
- Number of replenishments between forward and reserve areas
- Number of times when a pick is made from forward and reserve areas due to item shortage in the forward area
- Total saving

Various experiments are carried out for both Order-based and Slot-based Dynamic Algorithms. The experiments were coded with Java under the Microsoft Visual J++ Integrated Development Environment (IDE) and executed on a HP x 4000 workstation with Intel Pentium 4, 2 GHz processor and 1 GB RAM. Execution time was approximately 2 hours.

7. EXPERIMENTAL RESULTS

In this chapter, the performance of Slot-based and Order-based Dynamic Algorithms is compared for various warehouse settings.

Total saving, number of forward picks, replenishment between reserve and forward areas and stock outs are used as performance measures. We especially focus on total saving as a performance criterion:

$$\text{Total Saving} = (\text{saving}) \times (\text{forward picks}) - (\text{cost}) \times \left(\frac{\text{flow}}{\text{volume in the forward area}} \right)$$

Table 7.1 shows the parameters used in Slot-based Dynamic Algorithm experiments.

Parameter	Values
M, the initial order value	1000, 2000, 3000
k, the look ahead order value	200, 500, 1000, 2000, 3000, 5000, 8000
w, item unit volume (feet cube)	0.349, 0.25, 0.10, 0.07
saving, forward pick saving	0.25, 0.5
cost, cost of each replenishment trip	1.5, 4
s, number of slots	80, 135, 250, 350, 500, 1000, 2000, 4000
v, slot volume (feet cube)	72.38, 42.89, 23.16, 16.54, 11.58, 5.79, 2.9, 1.45

Table 7.1.Parameters in a warehouse

Parameters used in Order-based Dynamic Algorithm are depicted in Table 7.2.

Parameter	Values
M, the initial order value	1000, 2000, 3000
k, the look ahead order value	200, 500, 1000, 2000, 3000, 5000, 8000
w, item unit volume (feet cube)	0.349, 0.25, 0.10, 0.07
saving, forward pick saving	0.25, 0.5
cost, cost of each replenishment trip	1.5, 4
s, number of slots	80, 135, 250, 350, 500, 1000, 2000, 4000
v, slot volume (feet cube)	72.38, 42.89, 23.16, 16.54, 11.58, 5.79, 2.9, 1.45
o, order cycle	**100, 500, 1000, 2000, 3000**

Table 7.2.Parameters in a warehouse

Before comparing two algorithms' performance, each algorithm's performance is observed for changing warehouse designs. Section 7.1 investigates Slot-based Dynamic Algorithm.

7.1. Slot-based Dynamic Algorithm

Figure 7.1 illustrates the changes in the total saving as the number of slots in the forward area increases. Our warehouse settings are defined as M=1000, w=0.349, *saving*=0.25 and *cost*=1.5. Under these settings, the total saving decreases as the look ahead order values increase for 8 different slot values in the forward area. The reason for decrease is the rapidly diminishing number of picks. 80 slot in the forward area usually gives better results for large items (w=0.349, w=0.25) in all look ahead order value. On the other hand, as the item volume decreases (w=0.1, w=0.07), the advantage of using 80 slot decreases, generally 250, 350 or 500 slot gives superior results. This is observed in all experiments under various warehouse settings and one of them is illustrated in Figure 7.2.

In all the following figures, M indicates initial order number, w is unit volume, *saving* is the forward pick saving, *cost* is the replenishment cost between forward and reserve areas and k is the look ahead order value.

.

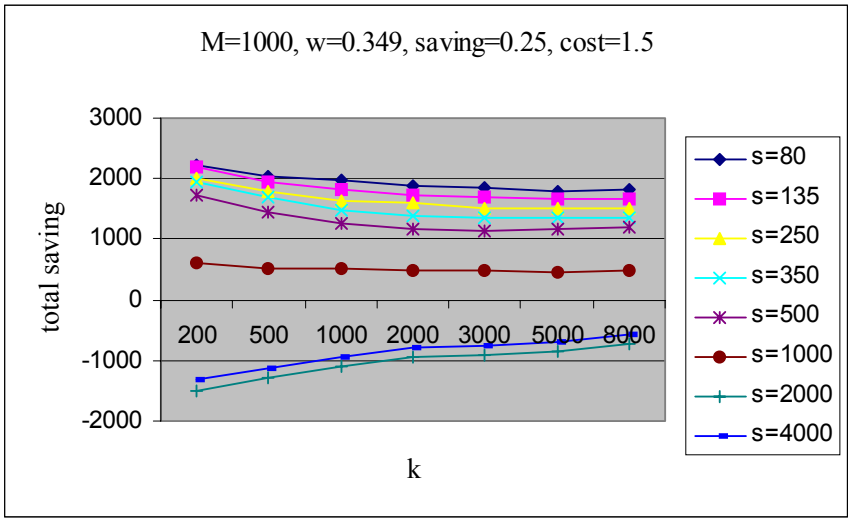


Figure 7.1. Comparison of total savings (*M*=1000, *w*=0.349, *saving*=0.25, *cost*=1.5)

Another interesting issue is the difference of total saving in Figure 7.1 and 7.2 for different slot values. For large items (*w*=0.349, *w*=0.25), Slot-based Dynamic Algorithm gives less total saving than for small items (*w*=0.1, *w*=0.07). Total saving begins to increase as the item volume decreases. The following reasoning can be carried out to explain this pattern:

As the item volume decreases, more item types can be stored in the forward slots. Therefore, our chance of finding and picking the demanded items from forward slots increases.

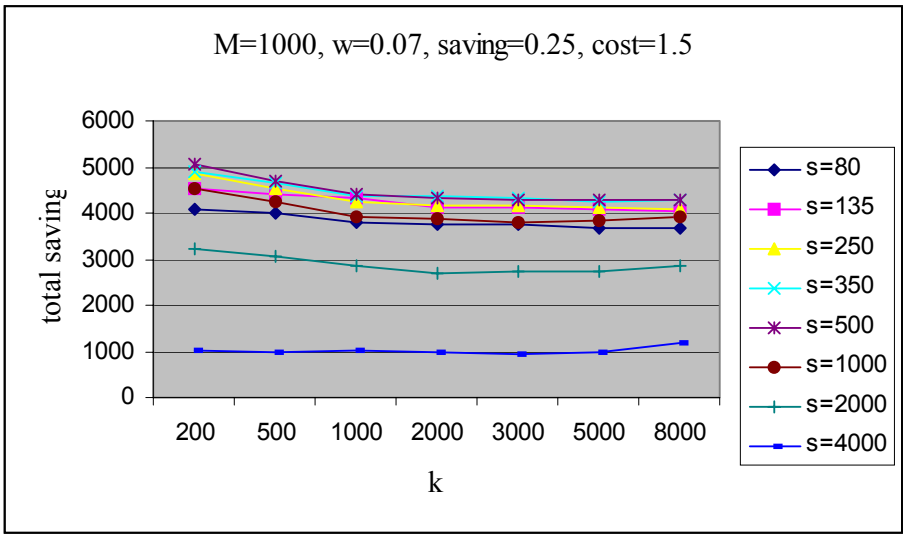


Figure 7.2. Comparison of total savings (*M*=1000, *w*=0.07, *saving*=0.25, *cost*=1.5)

As observed in Figure 7.3, increasing saving and cost parameters do not affect the savings data patterns. In this experiment, k=200 beats others in terms of saving.

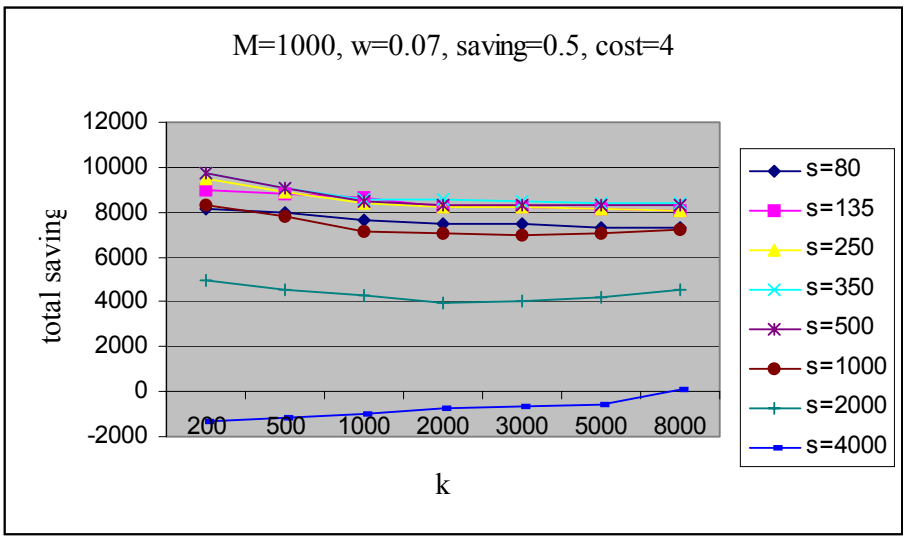


Figure 7.3. Comparison of total savings (*M*=1000, *w*=0.07, *saving*=0.5, *cost*=4)

If the initial order value, *M* is changed from 1000 to 2000 and 3000 respectively, as depicted in Figure 7.4 and 7.5, k=200 gives better saving for each number of slot in the forward pick area.

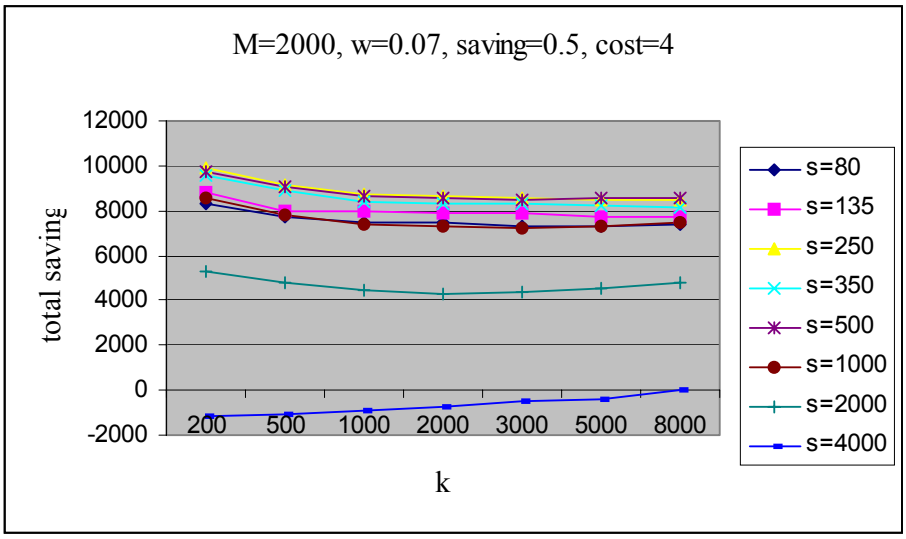


Figure 7.4. Comparison of total savings (*M*=2000, *w*=0.07, *saving*=0.5, *cost*=4)

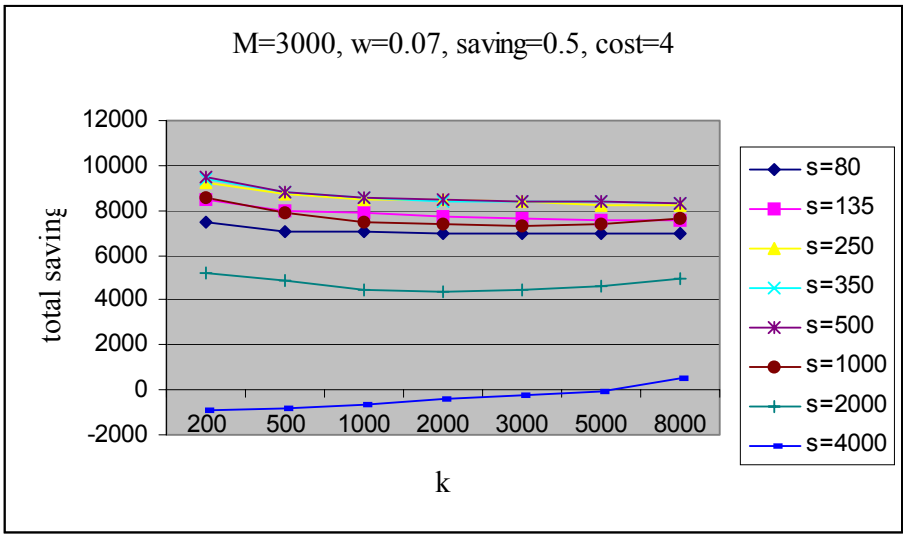


Figure 7.5. Comparison of total savings (*M*=3000, *w*=0.07, *saving*=0.5, *cost*=4)

In the Slot-based Dynamic Algorithm, using 80 slots in the forward area and observing k=200 for new item replenishment brings more savings for items whose volume is 0.349 and 0.25. On the other hand, for small items (*w*=0.1 and 0.07), using 250, 350 or 500 slot in the forward area and having k=200 is more profitable.

In Figure 7.6 and 7.7 the number of forward picks for items with large volume (*w*=0.349) and small volume (*w*=0.07) is observed under the same warehouse settings

(*M*=1000, *saving* =0.25 and *cost*=1.5). The number of picks increases as the number of slots increases. For large items (*w*=0.349), generally 1000 or 2000 slots in the forward area give maximum pick numbers and number of forward pick begins to decrease approximately after 1000 to 2000 slots. In all the experiments, k=200 gives the best values for forward pick.

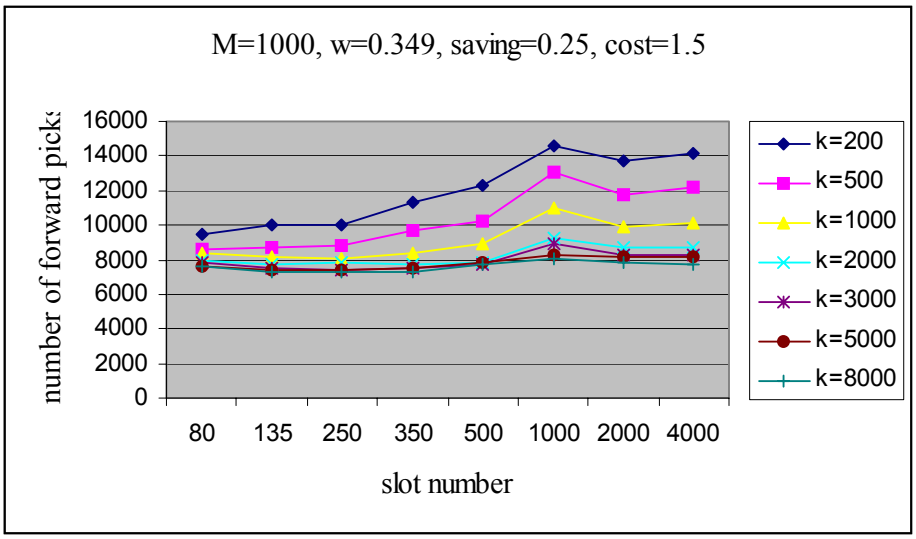


Figure 7.6. Comparison of number of forward picks (*M*=1000, *w*=0.349, *saving*=0.25, *cost*=1.5)

As it is seen from Figure 7.7, number of picks for small items always increases while the number of slots in the forward area increases. It should be noted that storing small items in forward slots create more item diversity and increase overall picking chance. Therefore, the forward picks in Figure 7.7 overwhelm the ones in Figure 7.6.

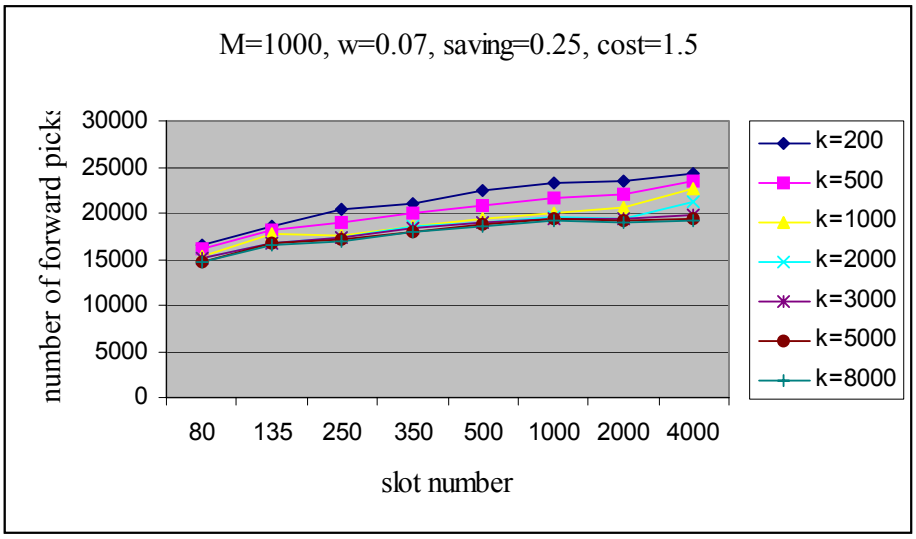


Figure 7.7. Comparison of number of forward picks (*M*=1000, *w*=0.07, *saving*=0.25, *cost*=1.5)

If items are large and Slot-based Dynamic Algorithm is applied, more slot than 2000 in the forward area does not increase picking chance. For small items, it is obvious that more slots in the forward area increase picking activity.

The number of replenishments under the warehouse settings *M*=1000, *saving*=0.25 and *cost*=1.5 for small and large items are depicted in Figure 7.8 and 7.9 respectively.

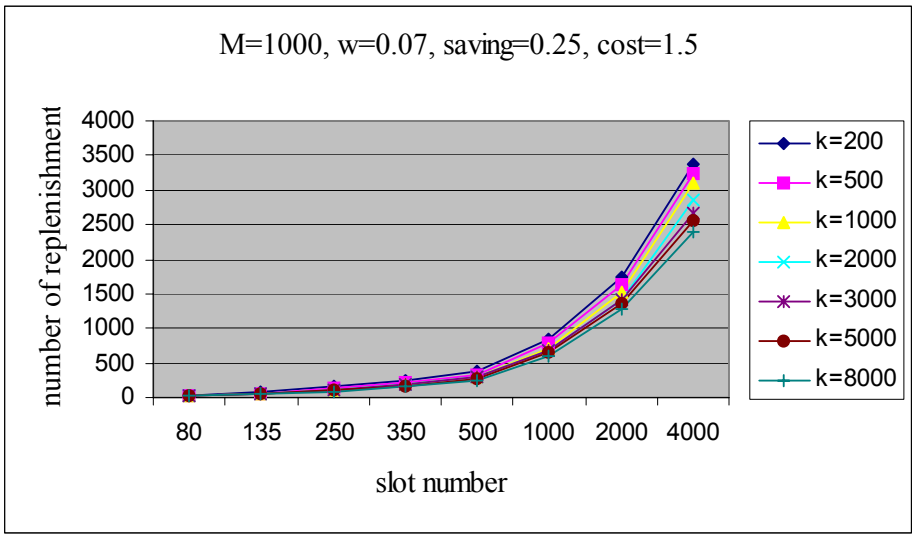


Figure 7.8. Comparison of number of replenishments (*M*=1000, *w*=0.07, *saving*=0.25, *cost*=1.5)

The number of replenishments for small items (*w*=0.07) always increases with the increasing number of slot in the forward area. Especially after 500 slots, replenishment activity begins to speed up.

For large items (*w*=0.349), number of replenishments increases with the increasing number of slot, but it generally decreases between 2000 and 4000 slots. The minimum replenishment cost is obtained when there are 80 slots in the forward pick area.

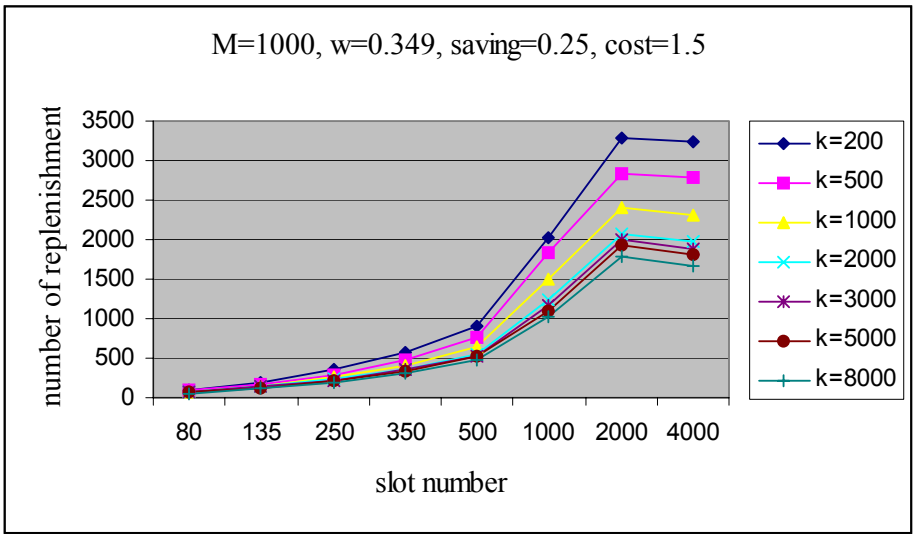


Figure 7.9. Comparison of number of replenishments (*M*=1000, *w*=0.349, *saving*=0.25, *cost*=1.5)

Finally, the stock out under the warehouse settings *M*=1000, *saving*=0.25 and *cost*=1.5 for large and small items are depicted in Figure 7.10 and 7.11 respectively. The number of stock outs generally increases with the increasing number of slots for all look ahead order value. As the item volume decreases, the number stock outs diminishes. The reason is that our chance for picking the demanded item from the forward slots increase.

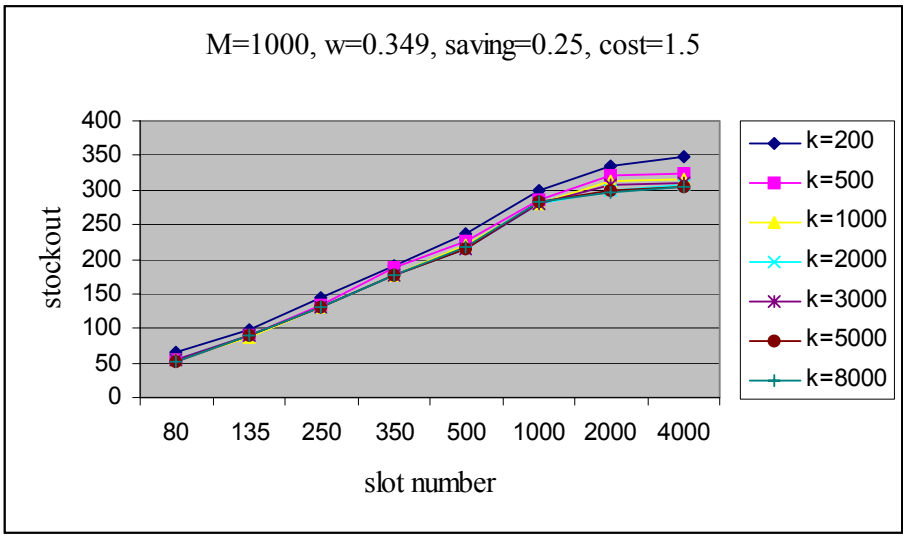


Figure 7.10. Comparison of number of stock out (*M*=1000, *w*=0.349, *saving*=0.25, *cost*=1.5)

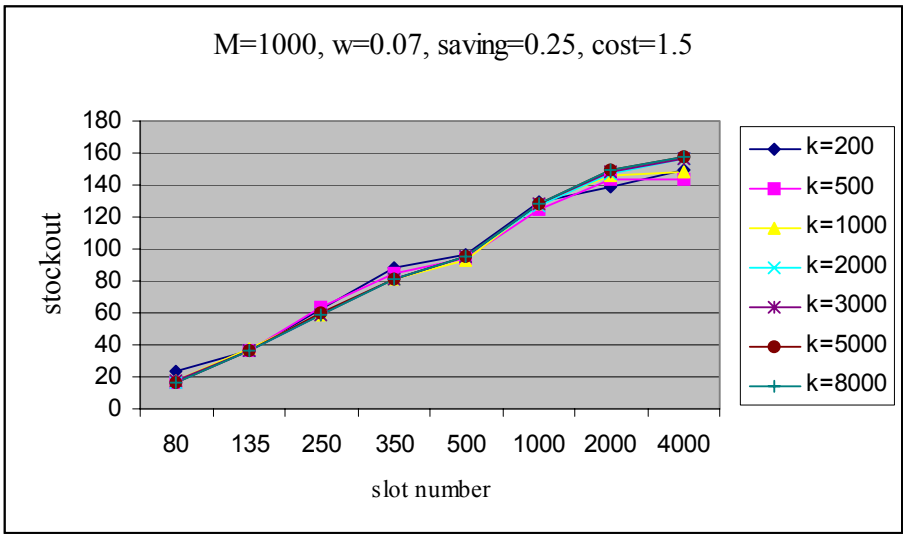


Figure 7.11. Comparison of number of stock out (*M*=1000, *w*=0.07, *saving*=0.25, *cost*=1.5)

In Appendix G, Slot-based Dynamic Algorithm experiments under different warehouse parameters are illustrated.

7.2. Order-based Dynamic Algorithm

In all the following figures, *M* is initial order number, *k* is look ahead order value, *w* is unit volume, *saving* is the forward pick saving, *cost* is the replenishment cost between forward and reserve areas and *o* is the order cycle.

In Figure 7.12, 7.13 and 7.14, it is seen that smallest order cycle 100 does not bring advantage as it is expected. A similar pattern is observed for warehouses with different parameters.

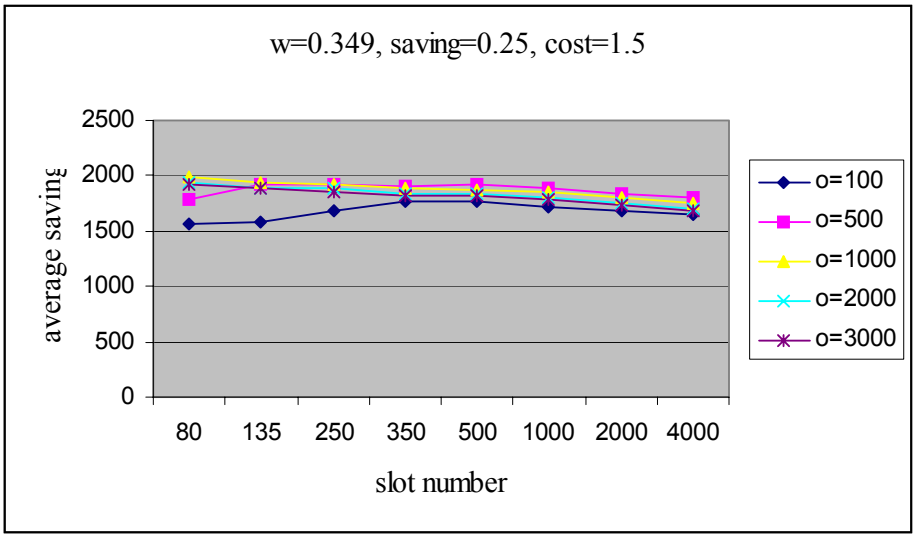


Figure 7.12. Comparison of total average savings (*w*=0.349, *saving*=0.25, *cost*=1.5)

For large items, there is a tendency for decrease in the total average saving as the number of slot increases in the forward pick area. On the contrary, a trend for increase is observed for small items.

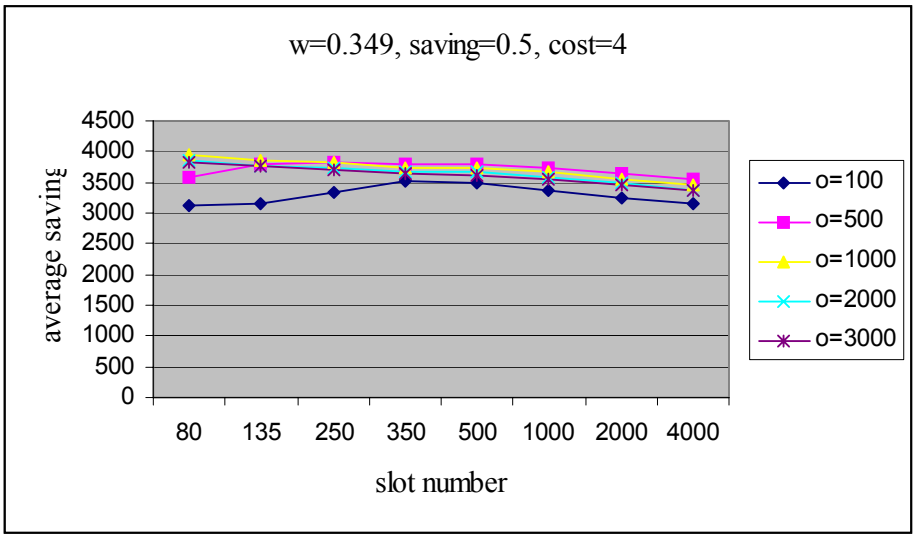


Figure 7.13. Comparison of total average savings (*w*=0.349, *saving*=0.5, *cost*=4)

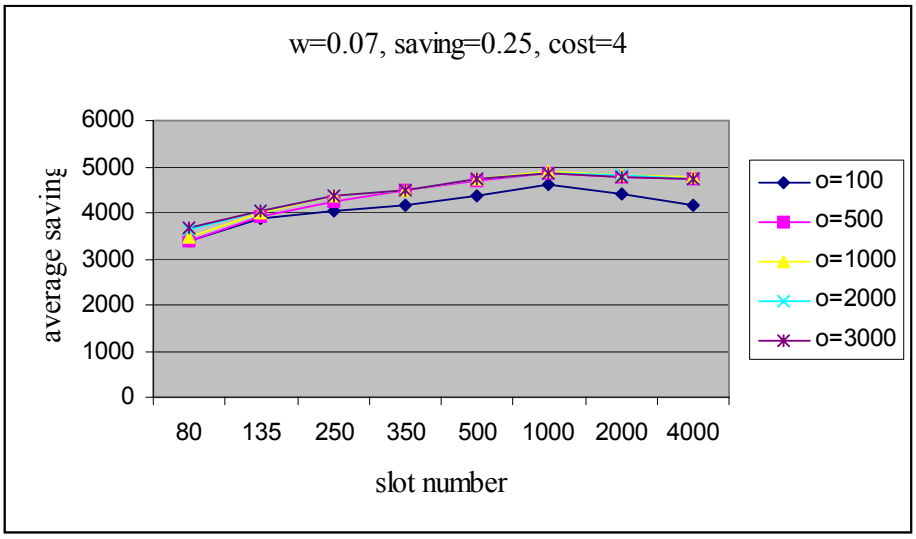


Figure 7.14. Comparison of total average savings (*w*=0.07, *saving*=0.25, *cost*=4)

In Figure 7.15, 7.16 and 7.17, the effect of look ahead order value is observed. k=200 generally gives better results. As k increases, number of forward picks decrease. The same pattern is also seen in the Slot-based Dynamic Algorithm.

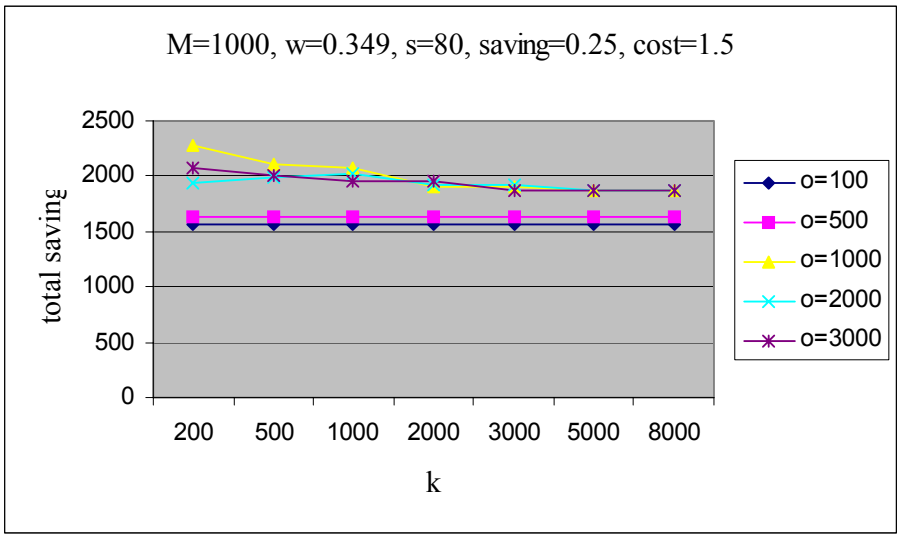


Figure 7.15. Comparison of total savings (*M*=1000, *w*=0.349, s=80, *saving*=0.25, *cost*=1.5)

When the item volume decreases from 0.349 to 0.07, the overall total saving increases. As we mentioned before, number of picks increase while the volume decreases.

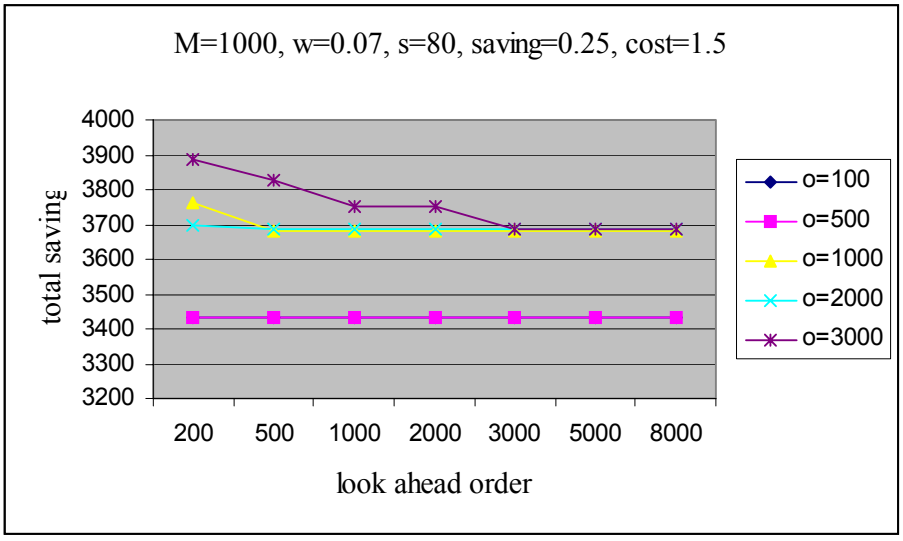


Figure 7.16. Comparison of total savings (*M*=1000, *w*=0.07, s=80, *saving*=0.25, *cost*=1.5)

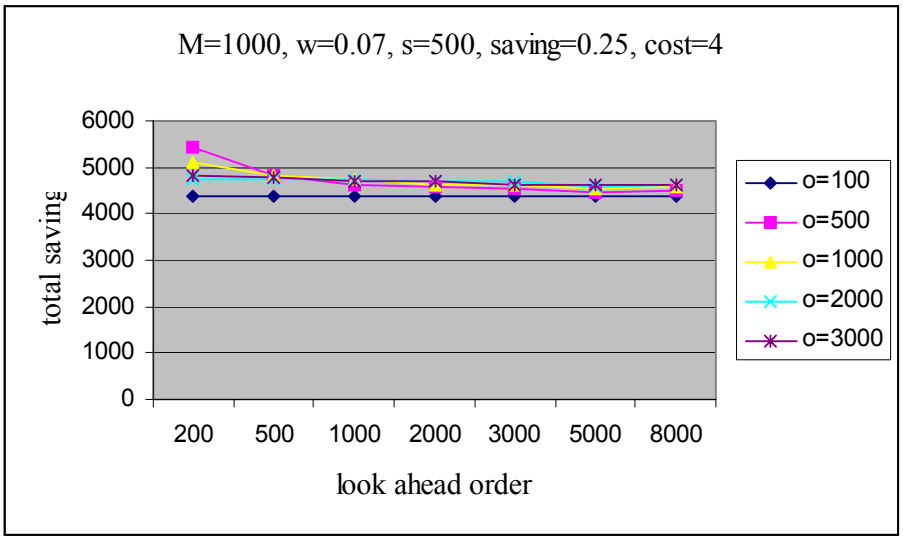


Figure 7.17. Comparison of total savings (*M*=1000, *w*=0.07, s=500, *saving*=0.25, *cost*=4)

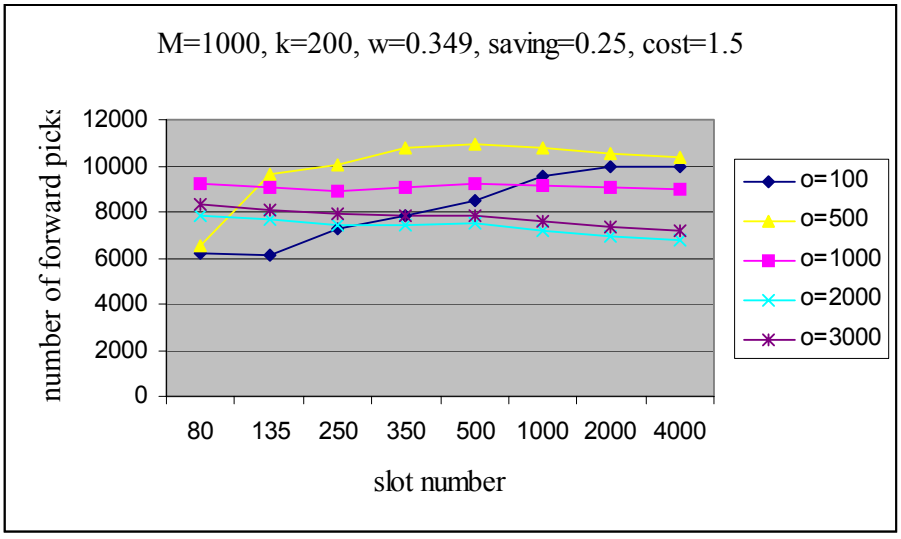


Figure 7.18. Comparison of forward picks (*M*=1000, *k*=200, *w*=0.349,*saving*=0.25, *cost*=1.5)

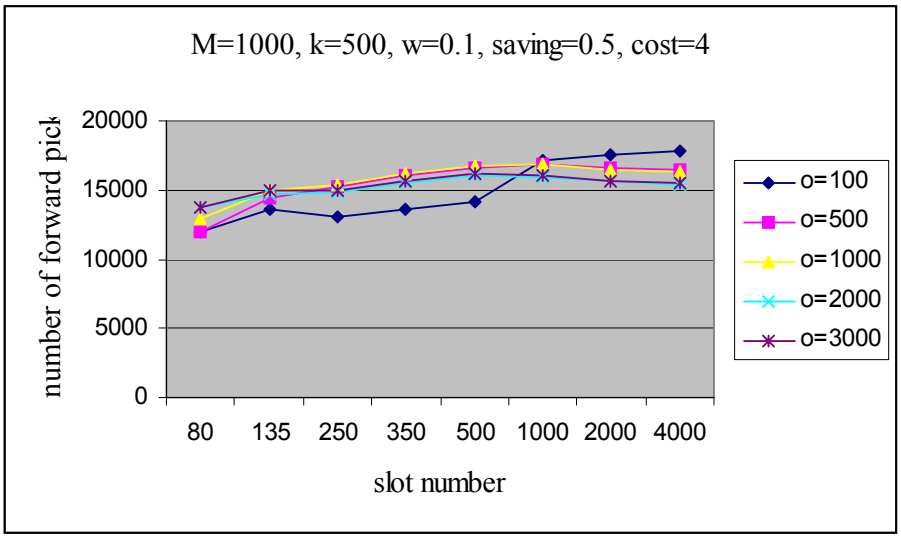


Figure 7.19. Comparison of forward picks (*M*=1000, *k*=500, *w*=0.1, *saving*=0.5, *cost*=4)

The volatile forward pick data pattern observed in Figure 7.18 and 7.19 occurs because of the Order-based Dynamic Algorithm logic. In the algorithm, the slots become empty until the end of the definite order cycle. Therefore, the demanded item might not be found in the forward area and picking activity is postponed. The same data pattern can be observed in replenishment activity. It is observed in all experiments that, o=100 does not bring the best forward pick value. Therefore, it is better to consider carefully about implementing Order-based_100 algorithm.

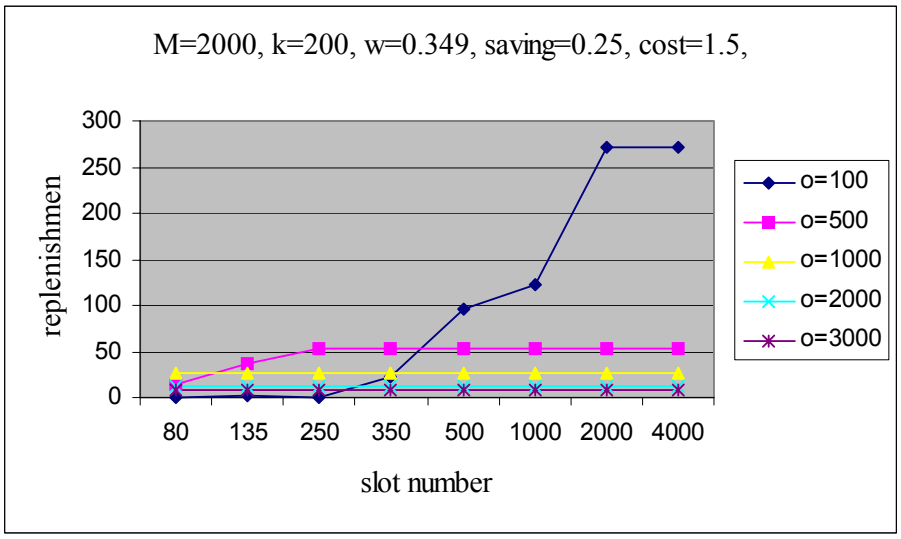


Figure 7.20. Comparison of replenishment (*M*=2000, *k*=200, *w*=0.349, *saving*=0.25 *cost*=1.5)

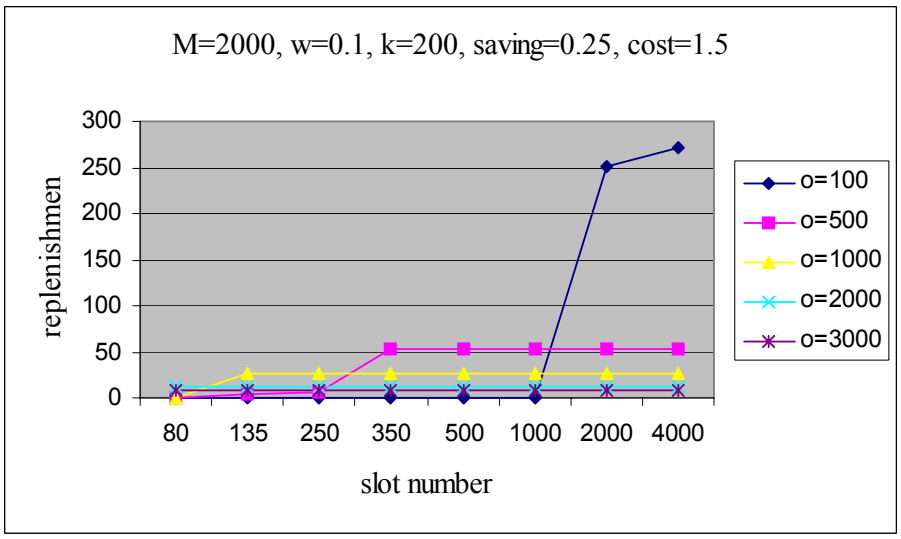


Figure 7.21. Comparison of replenishment (*M*=2000, *k*=200, *w*=0.1 *saving*=0.25 *cost*=1.5)

The stock out situation is observed in the following figures. All order cycles approximately give the same stock out for each slot value.

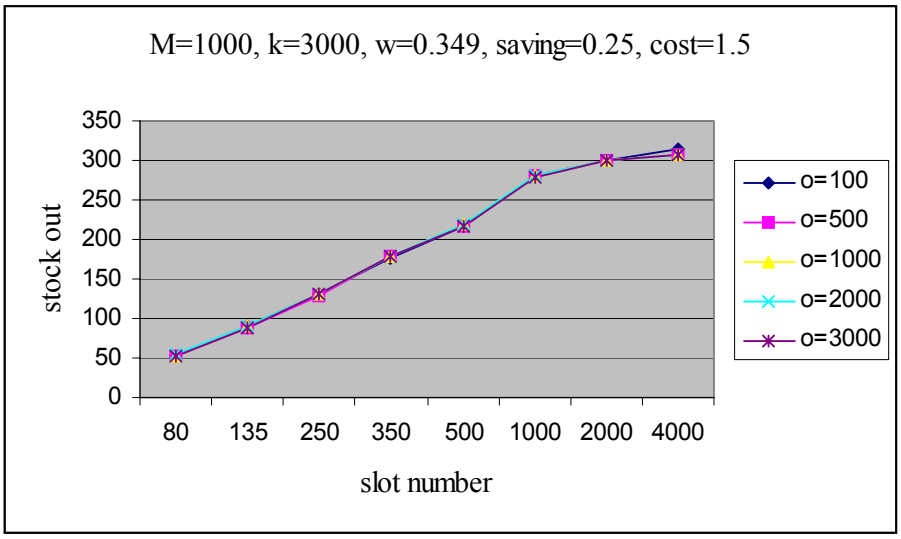


Figure 7.22. Comparison of stock out (*M*=1000, *k*=3000, *w*=0.349 *saving*=0.25 *cost*=1.5)

In Appendix H, Order-based Dynamic Algorithm experiments under different warehouse parameters are illustrated.

7.3. Slot-based and Order-based Dynamic Algorithms

Slot-based and Order-based Dynamic Algorithms' performance for various warehouse designs are compared in this section. In Table 7.3, 7.4, 7.5 and 7.6, average percentage difference in total saving of the Slot-based Dynamic Algorithm from the Order-based Dynamic Algorithms is examined under various warehouse settings.

In Table 7.3, 7.4, 7.5 and 7.6, negative (-) difference means that Order-based Algorithm gives better results than Slot-based Algorithm with the defined percentage and "x" indicates that applying Slot-based Algorithm ends with loss under the specific warehouse parameters. The distinguishing warehouse parameters are *saving* and *cost* values in Table 7.3, 7.4, 7.5 and 7.6

In all the following tables and figures, *M* is initial order number, *k* is look ahead order value, *w* is unit volume, *saving* is the forward pick saving, *cost* is the

replenishment cost between forward and reserve areas, *o* is the order cycle and *s* is the total number of slot in the forward pick area.

As observed from Table 7.3, 7.4, 7.5 and 7.6, Slot-based Algorithm gives superior average total saving than Order-based_100 and Order-based_500 for 80 slot in all different unit volume parameters (w=0.349, 0.25,0.1 and 0.07). Another important point is that, advantage of using Slot-based Algorithm decreases with the increasing number of slots because of rapidly growing replenishments. It starts to give insufficient average saving or complete loss especially after 500 slot in the forward pick area for all warehouse settings.

Slot-based Dynamic Algorithm is highly sensitive to changes in the cost and saving parameters that effect replenishment and pick values. Whereas, these changes generally have little influence in Order-based Algorithms.

If the cost and saving variables usually change, the decision makers have to be more careful about implementing Slot-based Algorithm.

saving=0.25, *cost*=1.5	*w*=0.349 Order Cycles					*w*=0.25 Order Cycles					*w*=0.1 Order Cycles					*w*=0.07 Order Cycles				
	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000
s=80	**24**	**8**	-2	-2	0	**20**	**15**	**5**	0	**1**	**12**	**12**	**6**	**1**	**1**	**10**	**10**	**7**	**2**	**1**
s=135	**17**	-4	-5	-3	-3	**14**	-3	-5	-3	-2	**8**	**2**	-1	0	0	**5**	**4**	**1**	0	0
s=250	**1**	-19	-12	-10	-9	**8**	-8	-9	-7	-6	**10**	-1	-3	-2	-1	**8**	**2**	-1	-1	0
s=350	-13	-19	-18	-16	-15	-5	-16	-15	-13	-12	**7**	-6	-5	-4	-4	**6**	-4	-4	-3	-3
s=500	-24	-30	-28	-27	-26	-20	-24	-24	-22	-21	**1**	-10	-9	-8	-8	**2**	-7	-7	-6	-5
s=1000	-63	-69	-66	-65	-65	-48	-52	-52	-50	-50	-19	-25	-24	-23	-23	-13	-18	-17	-17	-16
s=2000	x	x	x	x	x	x	x	x	x	x	-50	-52	-51	-51	-51	-38	-40	-40	-39	-.39
s=4000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	-77	-78	-77	-77	-77

Table 7.3. Average % difference in total savings (*saving*=0.25, *cost*=1.5).

The positive values are shown in bold.

saving=0.5, *cost*=4	*w*=0.349 Order Cycles					*w*=0.25 Order Cycles					*w*=0.1 Order Cycles					*w*=0.07 Order Cycles				
	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000
s=80	**22**	**6**	-4	-2	-1	**19**	**14**	**4**	-1	0	**12**	**12**	**6**	0	0	**10**	**10**	**7**	**1**	0
s=135	**12**	-8	-9	-7	-7	**12**	-5	-6	-5	-4	**7**	**2**	-1	-1	-1	**5**	**4**	0	0	0
s=250	-6	-18	-18	-16	-15	**3**	-12	-12	-11	-10	**8**	-3	-4	-3	-3	**6**	**1**	-3	-2	-2
s=350	-23	-28	-28	-26	-25	-12	-21	-20	-19	-18	**4**	-8	-8	-7	-6	**4**	-5	-5	-5	-4
s=500	-38	-43	-42	-41	-40	-28	-32	-32	-31	-30	-3	-13	-13	-12	-12	-1	-9	-9	-8	-8
s=1000	x	x	x	x	x	-69	-72	-72	-71	-71	-28	-33	-33	-32	-32	-20	-24	-24	-23	-23
s=2000	x	x	x	x	x	x	x	x	x	x	-68	-70	-70	-69	-69	-51	-53	-53	-53	-53
s=4000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 7.4. Average % difference in total savings (*saving*=0.5, *cost*=4).

The positive values are shown in bold.

saving=0.25, *cost*=4	*w*=0.349 Order Cycles					*w*=0.25 Order Cycles					*w*=0.1 Order Cycles					*w*=0.07 Order Cycles				
	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000
s=80	**14**	0	-8	-7	-7	**14**	**10**	0	-4	-3	**10**	**10**	**4**	-1	-1	**8**	**8**	**6**	**1**	0
s=135	-3	-18	-19	-19	-18	**2**	-12	-13	-13	-12	**3**	-2	-3	-3	-4	**2**	**1**	-1	-2	-2
s=250	-29	-37	-37	-36	-36	-15	-25	-45	-26	-26	0	-8	-9	-9	-9	**1**	-3	-6	-6	-6
s=350	-57	-60	-60	-59	-59	-36	-42	-42	-41	-41	-8	-16	-17	-17	-17	-4	-11	-12	-12	-12
s=500	x	x	x	x	x	-68	-69	-69	-69	-69	-21	-28	-28	-28	-28	-13	-19	-20	-20	-20
s=1000	x	x	x	x	x	x	x	x	x	x	-65	-68	-68	-68	-68	-47	-49	-49	-49	-49
s=2000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
s=4000	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 7.5. Average % difference in total savings (*saving*=0.25, *cost*=4)

The positive values are shown in bold.

saving=0.5, *cost*=1.5	*w*=0.349 Order Cycles					*w*=0.25 Order Cycles					*w*=0.1 Order Cycles					*w*=0.07 Order Cycles				
	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000	100	500	1000	2000	3000
s=80	**27**	**9**	-1	**2**	**3**	**22**	**17**	**6**	**1**	**2**	**13**	**13**	**7**	**1**	**1**	**10**	**10**	**8**	**2**	**1**
s=135	**21**	-1	-3	0	0	**18**	0	-2	0	**2**	**9**	**4**	0	**1**	**1**	**6**	**5**	**2**	**1**	**1**
s=250	**8**	-7	-6	-3	-2	**15**	-3	-4	-1	0	**12**	**1**	-1	**1**	**1**	**10**	**3**	0	**1**	**1**
s=350	-3	-11	-9	-6	-5	**4**	-8	-7	-4	-3	**11**	-2	-2	0	0	**9**	-1	-1	0	0
s=500	-10	-17	-15	-12	-11	-6	-12	-10	-8	-7	**8**	-5	-4	-2	-2	**8**	-3	-2	-1	-1
s=1000	-28	-33	-31	-29	-28	-21	-26	-25	-23	-22	-6	-12	-11	-9	-9	-3	-9	-8	-7	-6
s=2000	-59	-62	-61	-59	-59	-47	-50	-49	-47	-47	-24	-29	-25	-24	-23	-18	-20	-19	-18	-17
s=4000	x	x	x	x	X	-98	-98	-98	-98	-98	-48	-48	-48	-47	-47	-38	-38	-38	-37	-37

Table 7.6. Average % difference in total savings (*saving*=0.5, *cost*=1.5)

The positive values are shown in bold.

8. CONCLUSION AND FUTURE WORK

This thesis focuses on forward reserve allocation in a dynamic environment with discrete storage spaces in the forward area. The major objective of this research is to present profitable and dynamic forward allocations under various warehouse settings.

In Chapter 5, the two proposed algorithms, Slot-based and Order-based Dynamic Algorithms, are explained. These algorithms are developed for generating forward configurations. Slot-based Dynamic Algorithm starts item replenishment from reserve to forward slots whenever slots become empty. On the other hand Order-based Dynamic Algorithm postpones item replenishment until the end of definite order cycle. Both algorithms are based on the idea of reassignment of most profitable items to the forward pick area.

For the purpose of this thesis, all the items in the forward area are assumed to be stored in identical slots. In the experiments, real order data, which consists of generally small orders, is used. The algorithms are processed under various warehouse designs. These experimental results and insights are crucial for the planner to select the robust algorithm in the competitive and volatile market conditions.

These algorithms can be compared in terms of several performance measures, number of forward picks, replenishment, number of stock out and total saving. We focus on especially total saving, major performance measure, in order to evaluate the algorithms. The following parameters were used in the experiments:

- Initial order : 1000, 2000, 3000
- Look ahead order: 200, 500, 1000, 2000, 3000, 5000, 8000
- Unit volume: 0.349, 0.25, 0.10, 0.07
- Saving: 0.25, 0.5

- Cost: 1.5, 4
- Number of slots: 80, 135, 250, 500, 1000, 2000, 4000
- Slot volume: 72.38, 42.89, 23.16, 16.54, 11.58, 5.79, 2.9, 1.45
- Order cycle: 100, 500, 1000, 2000, 3000

Slot-based Algorithm gives superior average total saving than Order-based_100 and Order-based_500 for 80 slot in all different unit volume parameters (w=0.349, 0.25,0.1 and 0.07). In other words, Slot-based Algorithm can be selected instead of Order-based_100 or Order-based_500 in warehouses with rather few slots in the forward area. But it should be noted that Order-based Algorithms with large order cycles give better savings.

Another interesting issue is the Order-based Algorithm with the smallest order cycle (o=100) performance. Although this algorithm is considered to be more suitable for the dynamic allocation environment (replenishments begin after shorter order cycle), it is observed that this algorithm does not bring advantage in terms of savings as expected.

Slot-based Dynamic Algorithm is more sensitive to changes in saving and cost parameters than Order-based Dynamic Algorithm. Whereas, these changes generally have little influence in Order-based Algorithms. Therefore, the decision for implementing Slot-based Dynamic Algorithm in a warehouse where cost and saving parameters are volatile should be considered more carefully.

In Slot-based and Order-based Dynamic Algorithm experiments, it is observed that, k=200 (smallest selected value) generally gives the best saving. It is proved that considering data too much into the future for item selection is not profitable, besides being time consuming.

This research can be extended in many ways:

- In our experiments, the forward area volume is the same for all slot values. Different warehouse volumes can be observed and the ideal warehouse volume can be investigated for both algorithms.

- Various algorithms and rules can be developed in this area. Different selection rules can be implemented.
- Multiple-forward area cases can be considered.
- This research is based on the small items allocation. Large item replenishment is not fully addressed in the literature.

9. APPENDICES

Appendix A : Code Validation

10 orders and 5 different item types (ACM1, HEWC, CCP9, BORE, and LIO4) are used in order to validate the Java code. Results of hand simulation and program are compared for both algorithms.

Orders:

ACM1 1 HEWC1
CCP9 1
BORE 1
LIO4 1 HEWC1
HEWC4
LIO4 2 ACM1 5 HEWC10
CCP9 1 BORE 6 ACM1 4 LIO4 6
ACM1 8 BORE 2
CCP9 2 LIO4 3 ACM1 5
ACM1 3 BORE 5 LIO4 2 HEWC10 CCP9 1

Initial Order Number (*M*): 5
Look Ahead Order Value (*k*): 2
Order Cycle (*o*): 2
Item Volume (*w*): 0.1 feet cube
Saving (*saving*):0.25
Cost (*cost*): 1.5
Number of slot (*s*):10
Slot Volume (*v*):1 feet cube

The Java code results are the following:

Slot-based Dynamic Algorithm:

Pick: 12
Replenishment: 2
Stock out: 1
Saving: 0

Order-based Dynamic Algorithm:

Pick: 12
Replenishment: 0
Stock out: 1
Saving: 3

Hand Simulation For Slot-based Dynamic Algorithm:

- Take the first 5 orders for initial allocation and then calculate the pick, flow and viscosity of these items:

$$Flow = Demand * Item\ Volume$$

$$Viscosity = \frac{pick}{\sqrt{flow}}$$

Item Type	Demand	Pick	Flow	Viscosity
ACM1	1	1	0.1 (0.1 feet cube*1)	**3.16**
HEWC	6	3	0.6 (0.1 feet cube*6)	3.87
CCP9	1	1	0.1 (0.1 feet cube*1)	3.16
BORE	1	1	0.1 (0.1 feet cube*1)	3.16
LIO4	1	1	0.1 (0.1 feet cube*1)	3.16

- After defining the items and their viscosities, all the items are sorted with respect to their viscosities:

HEWC – ACM1- CCP9 – BORE – LIO4

- Best item set is evaluated for the initial allocation. 5 different item sets are

created. Savings obtained by these item sets are calculated:

$$Saving = pick * saving - (\cos t) * \frac{flow}{forwardvolume}$$

Item Set	HEWC	HEWC-ACM1	HEWC-ACM1 CCP9	HEWC-ACM1 CCP9-BORE	**HEWC-ACM1 CCP9-BORE LIO4**
***forwardvolume* (feet cube)**	10	7.10-2.89	5.50-2.25-2.25	4.49-1.84-1.84-1.84	**3.80-1.55-1.55-1.55-1.55**
Saving	0.66	0.82	0.95	1.05	**3.797**

- It is obviously seen that, last item set is the most profitable set. This set is investigated for feasibility. The lower bound is checked:

$$lowerbound = \frac{\cos t * flow}{saving * pick}$$

Items	HEWC	ACM1	CCP9	BORE	LIO4
***forwardvolume* (feet cube)**	3.80	1.55	1.55	1.55	1.55
***lowerbound* (feet cube)**	1.2	0.6	0.6	0.6	0.6

Assigned volumes for all the items exceed lower bound constraints. Therefore, these items can be assigned to forward slots.

- The required number of slot is calculated. The forward volume is divided by slot volume and the results are rounded to the nearest integer:

Items	HEWC	ACM1	CCP9	BORE	LIO4
forwardvolume (feet cube)	3.80	1.55	1.55	1.55	1.55
slot	4	2	2	2	0
assignedvolume (feet cube)	4	2	2	2	0

The procedure for the initial allocation is the same in both algorithms, therefore, the simulation steps until here is also valid in Order-based Dynamic Algorithm.

- Simulation starts. Read orders after order 5:

Order 6:

Pick: 2 (Pick ACM1 and HEWC)
Saving: 0.5 (0.25*2)
Remaining Volume: HEWC 3 ACM1 1.5 CCP9 2 BORE 2
Check Slot: 1 slot is emptied (HEWC)
Decision: Select CCP9
Replenishment: 1
Cost: 1.5 (1.5*1)

Order 7:

Items: HEWC 3 ACM1 1.5 CCP9 3 BORE 2
Pick: 5 (Pick CCP9, BORE and ACM1)
Saving: 1.25 (0.25*5)
Remaining Volume: HEWC 3 ACM1 1.1 CCP9 2.9 BORE 1.4
Check Slot: There is no empty slot

Order 8:

Pick: 7 (Pick ACM1 and BORE)
Saving: 1.75 (0.25*7)
Remaining Volume: HEWC 3 ACM1 0.3 CCP9 2.9 BORE 1.2
Check Slot: 1 slot is emptied (ACM1)
Decision: Remain empty slot empty
Empty Slot: 1

Order 9:

Pick: 9 (Pick ACM1 and CCP9)
Saving: 2.25 (0.25*9)
Remaining Volume: HEWC 3 ACM1 0 CCP9 2.7 BORE 1.2
Stock out: 1 (ACM1)
Empty Slot: 2
Decision: Select CCP9
Replenishment: 2
Cost: 3 (1.5*2)

Order 10:

Items: HEWC 3 CCP9 4.7 BORE 1.2
Pick: 12 (Pick HEWC, CCP9 and BORE)
Saving: 3 (0.25*12)
Remaining Volume: HEWC 2 CCP9 4.6 BORE 0.7
Empty Slot: 2

- Simulation ends. Calculate total saving:

$$\text{Total saving} = \text{Saving} - \text{Cost}$$
$$= 3\text{-}3 = 0$$

Results of the hand simulation:

Pick: 12
Replenishment: 2
Stock out: 1
Saving: 0

The results of the hand simulation and the Java code are the same. Our code is validated.

Hand Simulation For Order-based Dynamic Algorithm:

- Start from the initial allocation. The items and their assigned volumes are illustrated below:

Items	HEWC	ACM1	CCP9	BORE
slot	4	2	2	2
assignedvolume (feet cube)	4	2	2	2

- Start simulation. Read orders after order5:

Order 6:

Order Cycle: 1
Pick: 2 (Pick ACM1 and HEWC)
Saving: 0.5 (0.25*2)
Remaining Volume: HEWC 3 ACM1 1.5 CCP9 2 BORE 2
Check Slot: 1 slot is emptied (HEWC)

Empty Slot: 1

Order 7:

Order Cycle: 2
Decision: Empty slot remains empty
Pick: 5 (Pick CCP9, BORE and ACM1)
Saving: 1.25 (0.25*5)
Remaining Volume: HEWC 3 ACM1 1.1 CCP9 1.9 BORE 1.4
Empty Slot: 1
Order Cycle: 0

Order 8:

Order Cycle: 1
Pick: 7 (Pick ACM1 and BORE)
Saving: 1.75 (0.25*7)
Remaining Volume: HEWC 3 ACM1 0.3 CCP9 1.9 BORE 1.2
Check Slot: 1 slot is emptied (ACM1)
Empty Slot: 2

Order 9:

Order Cycle: 2
Decision: There is not enough look ahead order value
Pick: 9 (Pick ACM1 and CCP9)
Saving: 2.25 (0.25*9)
Remaining Volume: HEWC 3 ACM1 0 CCP9 1.7 BORE 1.2
Stock out: 1 (ACM1)
Empty Slot: 3

Order 10:

Order Cycle: 0
Pick: 12 (Pick HEWC, CCP9 and BORE)
Saving: 3 (0.25*12)
Remaining Volume: HEWC 2 CCP9 1.6 BORE 0.7
Empty Slot: 4

- Simulation ends. Calculate total saving:

$$\text{Total saving} = \text{Saving} - \text{Cost}$$
$$= 3\text{-}0 = 3$$

Results of the hand simulation:

Pick: 12
Replenishment: 0
Stock out: 1
Saving: 3

The results of the hand simulation and the Java code are the same. Our code is validated.

Appendix B : Pseudo Code for the Initial Allocation Sub-Algorithm

In this sub-algorithm, the initial allocation of items in the forward area is determined for both Order-based and Slot-based Dynamic Algorithms.

Input :

- M : number of orders used in the initial allocation
- N : number of different item types in M orders
- *saving* : saving when a pick is made from the forward area
- *cost* : cost of each replenishment trip
- s : total number of slots in the forward area
- v : volume of each slot in the forward area
- TV : total volume of the forward area
- w : item volume

Variables:

- d : demand
- f : item flow

$$f = d * w$$

- p : item pick
- *viscosity* : item viscosity

$$vis\cos ity = \frac{p}{\sqrt{f}}$$

- u : volume assigned to item in the forward area

$$u = TV \ \frac{\sqrt{f}}{\sqrt{\sum f}}$$

- *lowerbound* : minimum item volume in the forward area

$$lowerbound = \frac{\cos t * f}{saving * p}$$

- *sn* : number of slot for each item
- *fv* : initial volume assigned to item in the initial allocation
- *TotalSavings* : total saving

$$TotalSavings = (saving * p) - (\cos t * \frac{f}{u})$$

Arrays:

- *Initialitem*[] : the array of *N* different item type
- *SortedItemArray*[] : the array of sorted items
- *bestItemArray* [] : the array of items considered for the initial allocation
- *forwardItemArray* [] : the array of items in the forward area

1. Calculate the flow, pick and viscosity values for all item types and sort items according to ascending viscosities:

Sort ***Initialitem*[]** by viscosity⟶ ***sortedItemArray*[]**

2. Determine the total saving of assigning first item to the forward area, first two items to the forward area and so on. Select the set that maximizes the total saving:

```
For (i=1 to N+1)
{
    For (n=0 to i)
    {
        Take sortedItemArray[n]
        Calculate TotalSavings
    }
```

}

/*Put in ***bestItemArray*[]** */

3. Calculate the assigned volume of each item. In the selected set, start with the last item, having the smallest viscosity, and check whether the item's assigned volume to the forward area exceeds the minimum volume or not. If it is below the minimum volume, the item is eliminated from the set:

```
For (i=0 to BN)                    /* BN is the length of bestItemArray[]*/
{
        Take bestItemArray[i]
        Calculate u
}
For (i=BN-1 to 0)
{
        Take bestItemArray[i]
        if (u< lowerbound)
        Remove from bestItemArray[]
}
BN=BN-1
```

4. Calculate the number of slot assigned to each item. The assigned volume is divided by slot volume and the result is rounded to next integer value. Therefore, the volume in the forward area is discretized:

```
For (i=0 to BN)
{
        Take bestItemArray[i]
        sn = Round (u / v)
        s = s-sn
}
```

5. Determine the initial volume assigned to each item in the forward pick area. Finally, create the set of items assigned to forward area.

```
For (i=0 to BN)
{
        Take bestItemArray[i]
        fv = sn*v
}
/* Put in ForwardItemArray[]*/
```

Appendix C: Pseudo Code for the Slot-based Dynamic Sub-Algorithm

Slot-based Dynamic Sub-Algorithm is implemented after Initial Allocation Sub-Algorithm. Initial Allocation Sub-Algorithm returns *ForwardItemArray*[], *sn* and *fv*. When slots in the forward area are emptied, a decision has to be made. The emptied slots may be filled with an item or may stay empty. Future order data is used as a decision criterion.

Input:

- *ForwardItemArray*[] : the array of items in the forward area
- *sn* : number of slot of each item
- *fv* : initial volume assigned to item in the initial allocation
- *k* : number of future orders used as a decision criterion
- *Orders* : number of orders in a period

Variables:

- *new_sn* : current number of slot for the item in the forward area after picking
- *empty_sn* : empty number of slot for the item
- *selecteditem* : the item replenished to the forward area and holds the empty slots
- *ordersize* : number of demanded items in an order
- *Forwardpicks* : number of forward picks
- *Replenishments* : number of replenishments between forward and reserve areas
- *Stockouts* : number of times when a pick has to be made from forward and reserve areas because of item shortage in the forward area.
- *Forwardsaving* : saving obtained by picking item from forward while there is a replenishment cost
- *Emptyslots* : number of empty slots in the forward area

Array:

- *item*[o] [j] : array of demanded items in an order, j=0,..,*ordersize*
  o=0,...*Orders*

1. Initialize the statistics:

Forwardpicks=0

Replenishments=0

Stockouts=0

Forwardsaving=0

Initialize the variables:

totalCost=0

totalSaving=0

2. Start from order *M* and read all the demanded items one by one in each order. If the demanded item exists in the forward area, pick the demanded item from the forward area and increase number of forward picks by one. During picking, three different situations might occur:

1. Flow of demanded item is smaller than the volume assigned to this item in the forward pick area. The volume assigned to the item in the forward area decrease and the new number of slot is determined.
2. Flow of demanded item is greater than the volume assigned to this item in the forward pick area. Stock out occurs and item is removed from the forward area.
3. Flow of demanded item is equal to the volume assigned to this item in the forward area. The item is removed from the forward area.

For (i = ***M*** to ***Orders***)

{

For (j=0 to ***ordersize***)

```
{
        Read item[i][j]
        if (item[i][j] ∈ ForwardItemArray[])
        {
        Calculate f
                Forwardpicks ++                 / *picking activity occurs*/
                totalSaving = totalSaving + saving
                 if ( f < fv)
                 {
                        fv=fv-f
                        new_sn= Round (fv / v)
                       GoTo Dynamic Decision (Step 3)
                 }
                 else if (f > fv)                        /* Stock out occurs*/
                 {
                       Stockouts ++            /*Increase stock out by one*/
                        fv=0
                        new_sn=0
                        Remove from ForwardItemArray[]
                        GoTo  Dynamic Decision (Step 3)
                 }
                 else
                 {
                        fv=0
                        new_sn=0
                        Remove from ForwardItemArray[]
                        GoTo  Dynamic Decision (Step 3)
                 }
            }
      }
}
```

3. Check whether slots become empty or not. Calculate empty slots. If the slots are emptied, then a dynamic decision is made:

```
if (sn-new_sn>0)            /* slots become empty*/
{
        empty_sn= new_sn-sn
        sn=new_sn
}
else
{
Return Step2                /* read the next item*/
}
```

4. In this step, the question of which items are assigned to empty slots is determined. We look ahead *k* orders. The flow and pick values of items in the *k* future order are calculated. Determine the total saving of assigning each item to the empty slots in the forward pick area. Select the item with the maximum total saving. If the saving is negative / zero then the empty slots remain empty:

```
MaxSaving=-999     /* Take the initial maximum saving a negative number*/
For (i=y to y+k)                    /* y is the current order*/
{
        For (j=0 to ordersize)
        {
                Take item[i][j]
                Calculate TotalSavings        /*Calculate the total saving */
                if(TotalSavings>MaxSaving)
                {
                MaxSaving=TotalSavings
                selecteditem=item[i][j]
                }
        }
```

}

if (***MaxSaving***>0) /* saving is obtained by assigning item to the empty slots*/

{

Replenishment ++ /* replenish the selected item to the forward area*/

totalCost = totalCost + cost /* replenishment cost occurs*/

Update ***ForwardItemArray*****[]**

/* if the selected item does not exist in the forward area, it is added to the array. Its number of slot becomes the empty slot*/

/* If the selected item does exist in the forward area, the new number of slot becomes the total of current and empty slot*/

}

else

{

/* The empty slots remain empty*/

}

5. Calculate the total saving:

Forwardsaving= Forwardsaving + (***totalSaving-totalCost***)

Appendix D : Pseudo Code for the Order-based Dynamic Sub-Algorithm

Order-based Dynamic Sub-Algorithm is implemented after Initial Allocation Sub-Algorithm. Initial Allocation Sub-Algorithm returns *ForwardItemArray*[], *sn* and *fv*. Empty slots in the forward area stay empty until the order cycle is finished. The empty slots are filled with the selected items. If the total saving of the selected item is negative / zero then the empty slots remain empty in the next order cycle. Like Slot-based Dynamic Sub-Algorithm, *k* is used as a decision criterion.

Input:

- *ForwardItemArray*[] : the array of items in the forward area
- *sn* : number of slot for each item
- *fv* : initial volume assigned to item in the initial allocation
- *o* : order cycle
- *k* : number of future orders used as a decision criterion
- *Orders* : number of orders in a period

Variables:

- *new_sn* : current number of slot of the item in the forward area after picking
- *empty_sn* : empty number of slot of the item
- *selecteditem* : the item replenished to the forward area and holds the empty slots
- *ordersize* : number of demanded items in an order
- *Forwardpicks* : number of forward picks
- *Replenishments:* number of replenishments between forward and reserve areas
- *Stockouts* : number of times when a pick has to be made from forward and reserve areas because of item shortage in the forward area
- *Forwardsaving* : saving obtained by picking item from forward while there is a replenishment cost
- *Emptyslots* : number of empty slots in the forward area

Array:

- *item*[o] [j] : array of demanded items in an order, j=0,..,*ordersize*
  o=0,...*Orders*

1. Initialize the statistics:

Forwardpicks=0

Replenishments=0

Stockouts=0

Forwardsaving=0

Initialize the variables:

totalCost=0

totalSaving=0

2. Start from order *M* and increase the order count index by one:

```
For (i = M to Orders)
{
    index ++            /*order count index is increased by one*/
      if ( index =o)     /* decision time…*/
      {
      GoTo  Replenishments (Step 3)
    }
    else
    {
      GoTo  Picking (Step 4)
    }
}
```

3. Order cycle is finished, now empty slots in the forward area are filled with the selected item. The flow and pick values of items in the *k* future order are calculated. Determine the total saving of assigning each item to the empty slots in the forward pick

area. Select the item with the maximum total saving:

```
MaxSaving=-999     /* Take the initial maximum saving a negative number*/
index=0
For (i=y to y+k)                    /* y is the current order*/
{
        For (j=0 to ordersize)
        {
                Take item[i][j]
                Calculate TotalSavings          /*Calculate the total saving */
                if(TotalSavings>MaxSaving)
                {
                MaxSaving=TotalSavings
                selecteditem=item[i][j]
                }
        }
}
if (MaxSaving>0)     /* saving is obtained by assigning item to the empty slots*/
{
        Replenishment  ++     /* replenish the selected item to the forward area*/
        totalCost = totalCost + cost       /* Replenishment cost occurs*/
        Update ForwardItemArray[]
}
else
{
/* The empty slots remain empty in the next order cycle*/
```

}

4. Read the items in each order. If the item exists in the forward area then pick the item. Update number of slot and assigned volumes:

```
For (j=0 to ordersize)
{
        Read item[i][j]

        if (item[i][j] ∈ ForwardItemArray[])
        {
        Calculate f
                Forwardpicks ++                 /* picking activity occurs*/
                totalSaving = totalSaving + saving

                if ( f < fv)
                {
                        fv=fv-f
                        new_sn= Round (fv / v)
                        empty_sn=sn-new_sn
                }
                else if (f > fv)                        /* Stock out occurs*/
                {
                       Stockouts ++             /*Increase stock out by one*/
                        fv=0
                        new_sn=0
                empty_sn=sn
                        Remove from ForwardItemArray[]
                }
                else
                {
                        fv=0
```

new_sn=0

empty_sn=sn

Remove from ***ForwardItemArray*[]**

}

}

else /* if the item does not exist in the forward pick area*/

{

Return Step 2

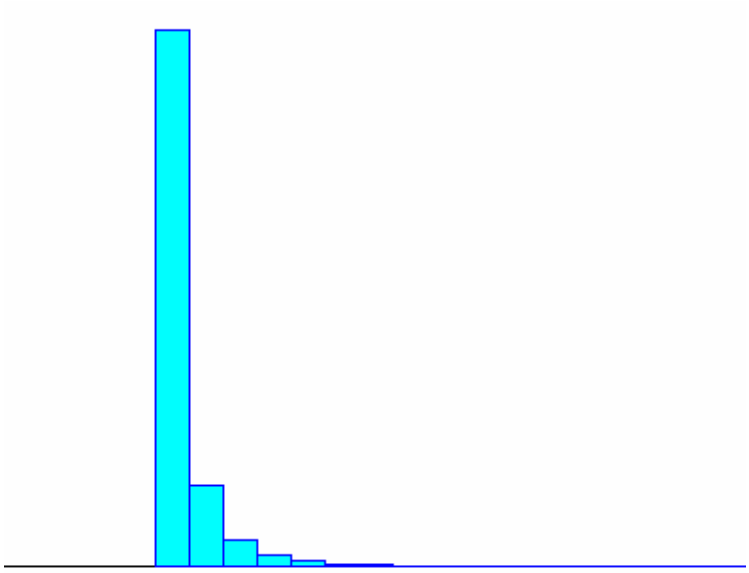
}

}

5. Calculate the total saving:

Forwardsaving= Forwardsaving + (***totalSaving-totalCost***)

Appendix E : The Histogram of Number of SKU Requests per Order



Histogram of number of SKU requests per order in S.P.R Data for 734 item type

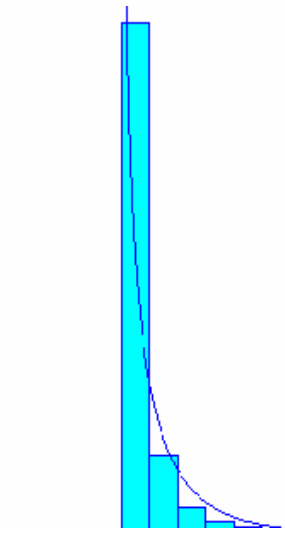
Data Summary

Number of Data Points = 29737
Min Data Value = 1
Max Data Value = 36
Sample Mean = 1.42
Sample Standard Deviation = 1.29

Histogram Summary

Histogram Range = 0.5 to 36.5
Number of Intervals = 36

Appendix F : The Distribution of Number of SKU Requests per Order



The distribution of number of SKU requests per order in S.P.R Data for 734 item type

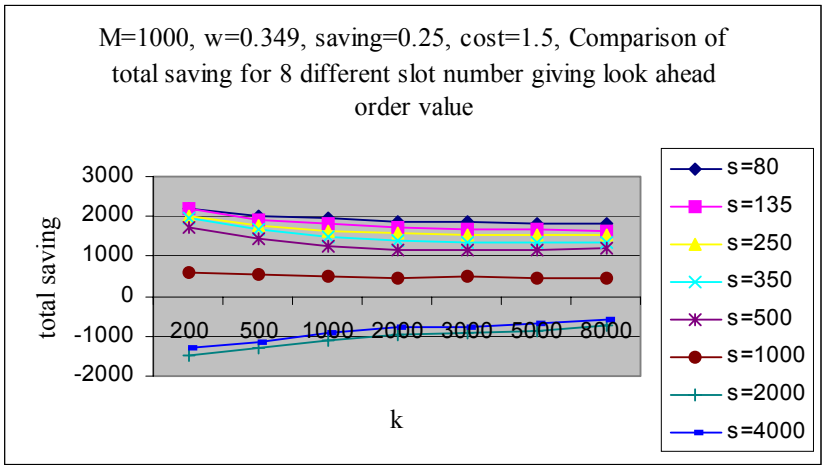
Distribution Summary

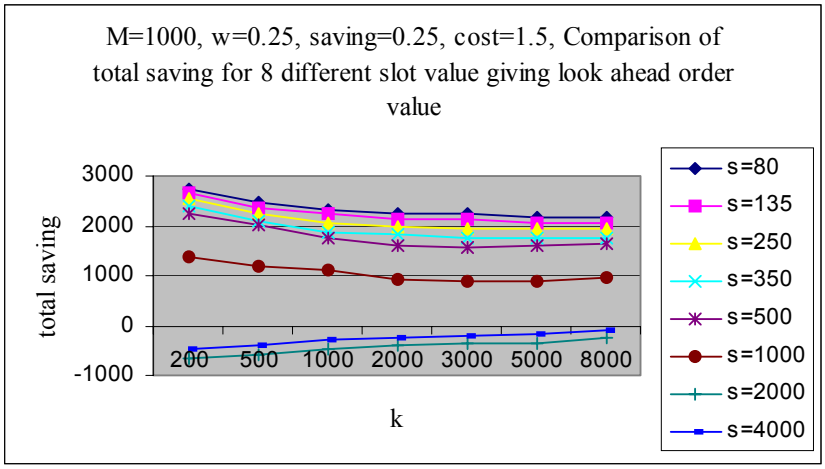
Distribution: Beta
Expression: 0.5 + 36 * BETA(0.464, 17.8)
Square Error: 0.01021

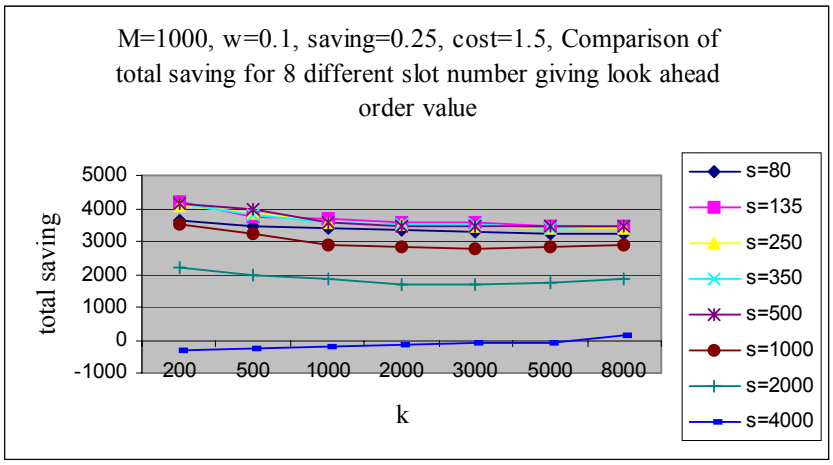
Chi Square Test

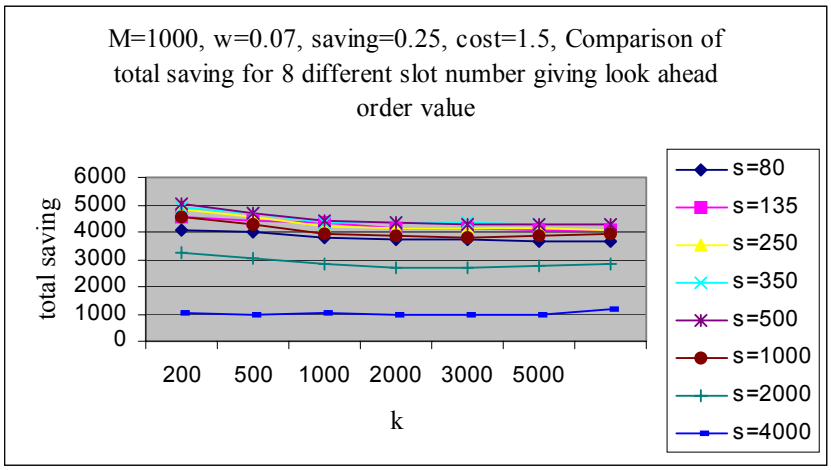
Number of intervals = 11
Degrees of freedom = 8
Test Statistic = 1.84e+003
Corresponding p-value < 0.005

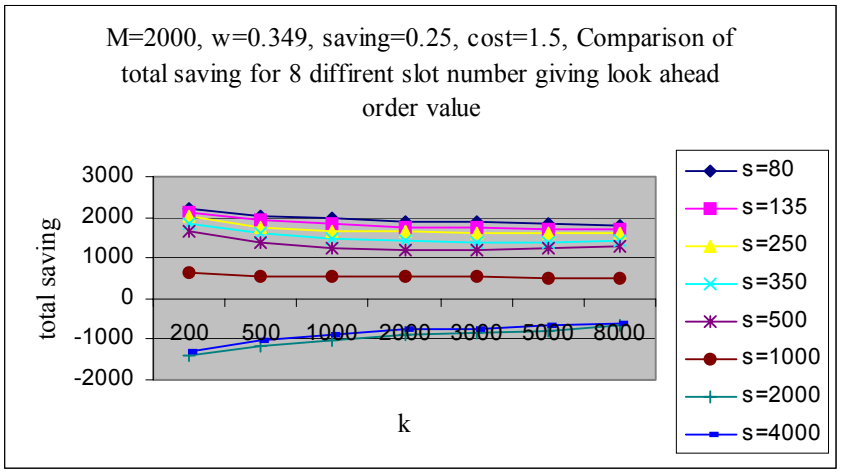
Appendix G : Slot-based Dynamic Algorithm Experiments

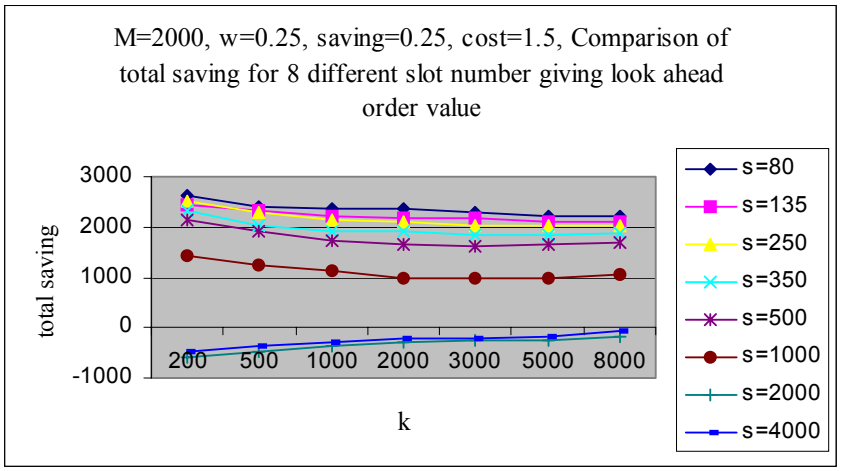


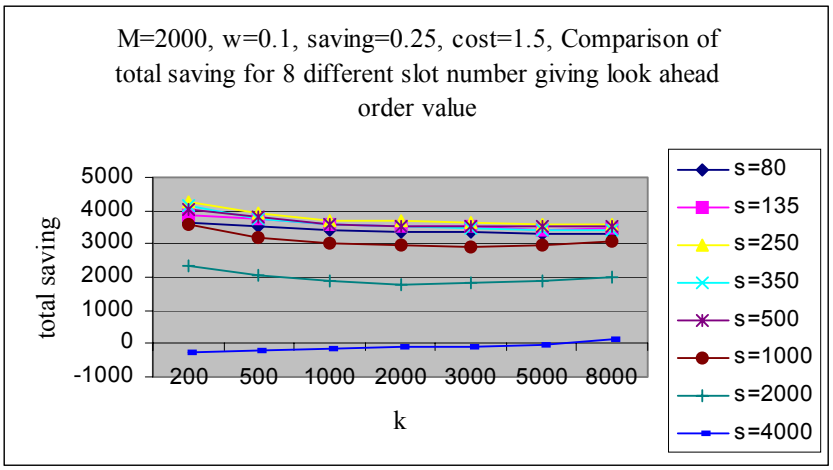


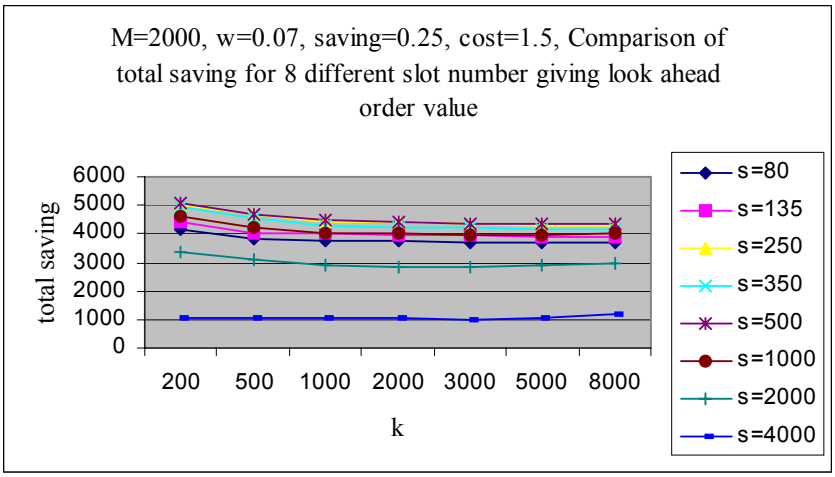


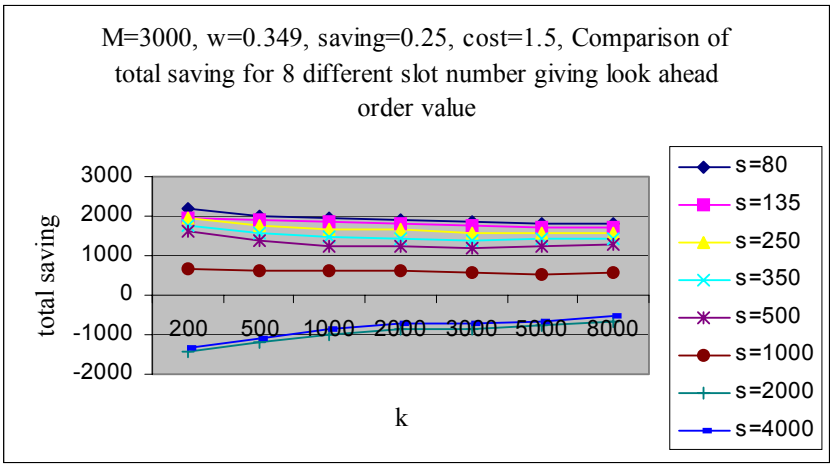

M=1000, w=0.07, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
5000
4000
3000
2000
1000
0
200
500
1000
2000
3000
5000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

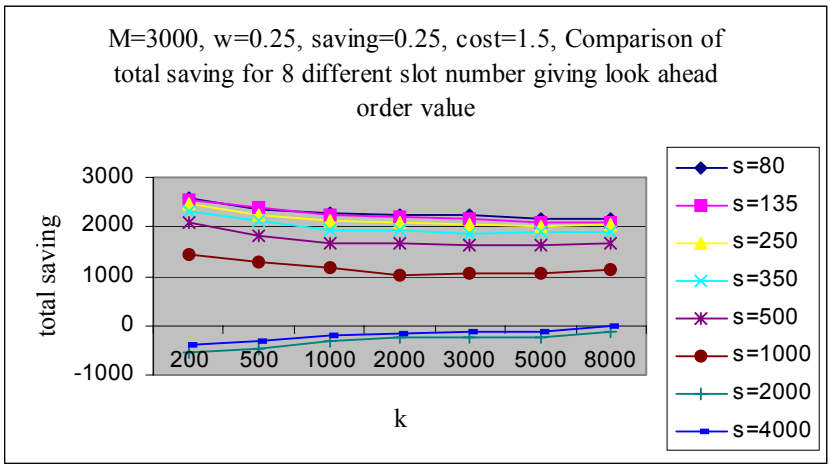

M=2000, w=0.349, saving=0.25, cost=1.5, Comparison of total saving for 8 diffirent slot number giving look ahead order value
total saving
3000
2000
1000
0
-1000
-2000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

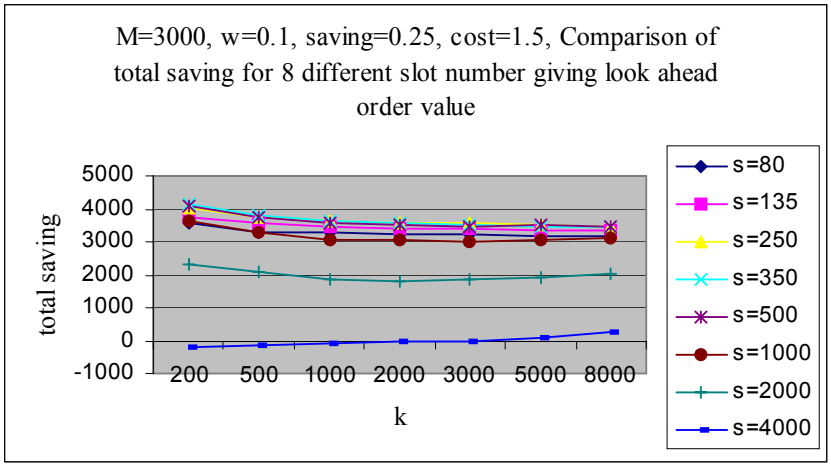

M=2000, w=0.25, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
3000
2000
1000
0
-1000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

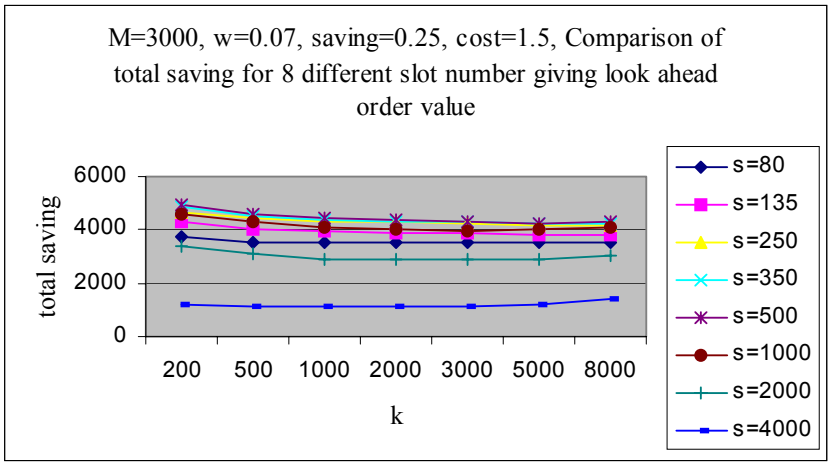

M=2000, w=0.1, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
5000
4000
3000
2000
1000
0
-1000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

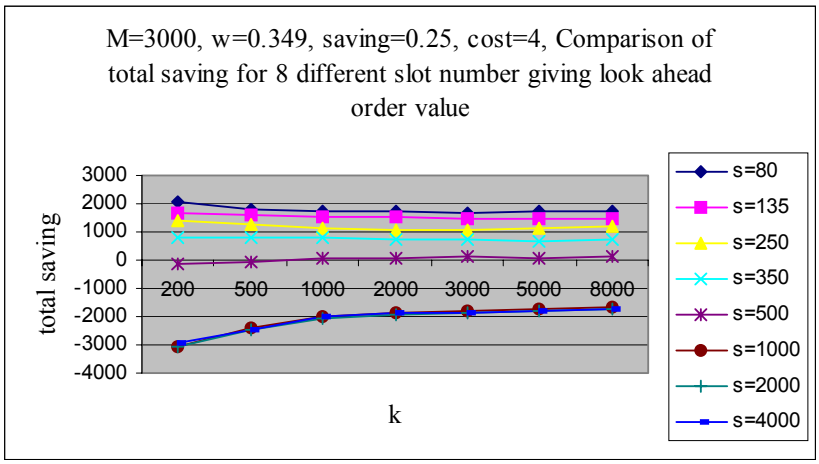

M=2000, w=0.07, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
5000
4000
3000
2000
1000
0
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

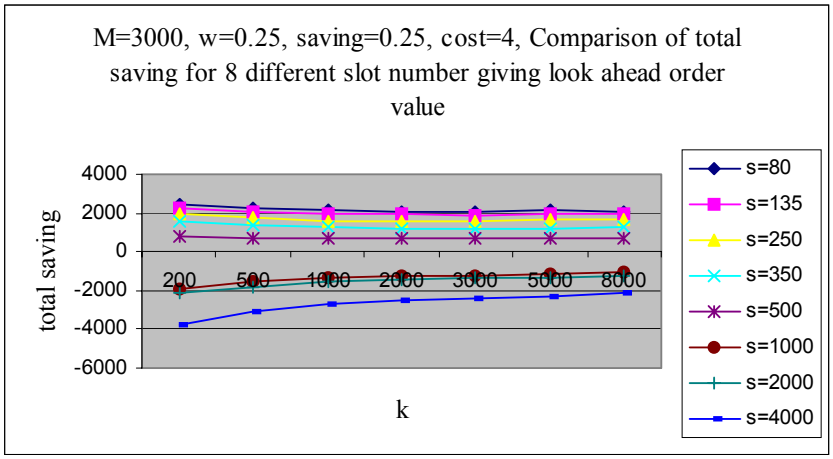

M=3000, w=0.349, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
3000
2000
1000
0
-1000
-2000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

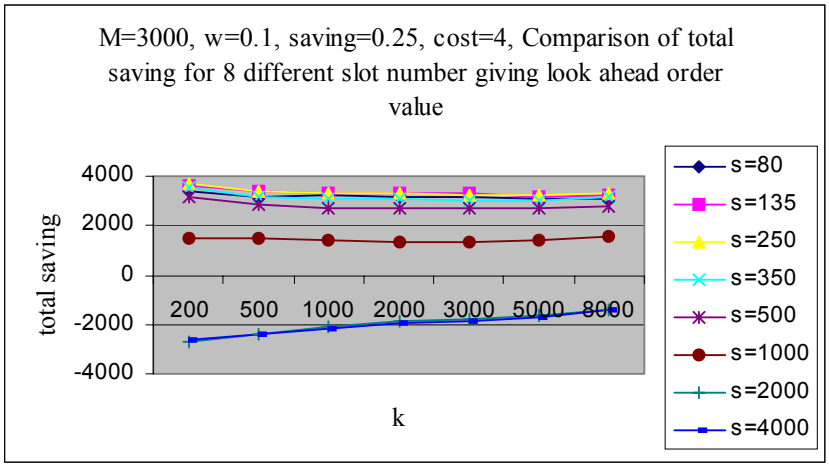

M=3000, w=0.25, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
3000
2000
1000
0
-1000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

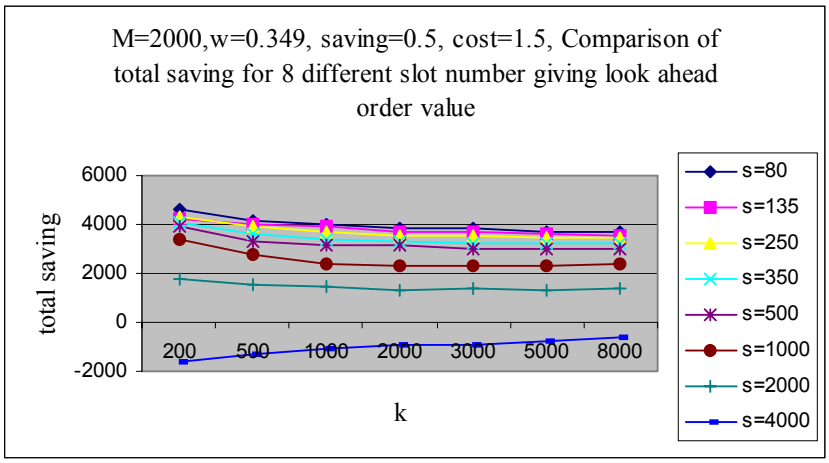

M=3000, w=0.1, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
5000
4000
3000
2000
1000
0
-1000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

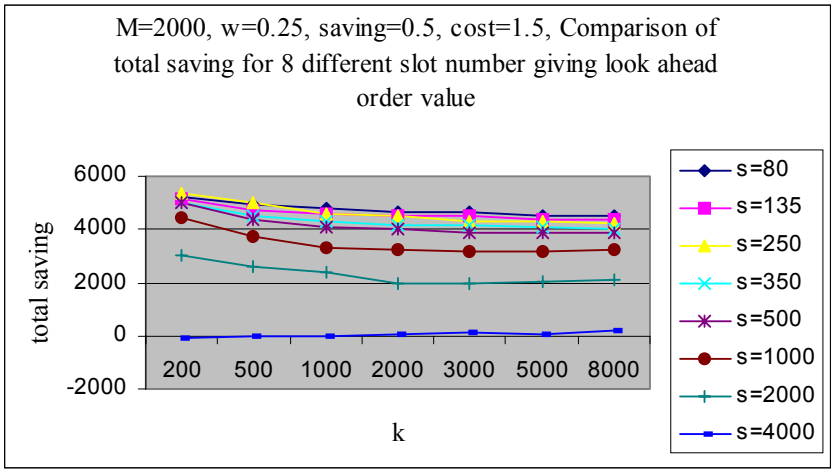

M=3000, w=0.07, saving=0.25, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
4000
2000
0
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

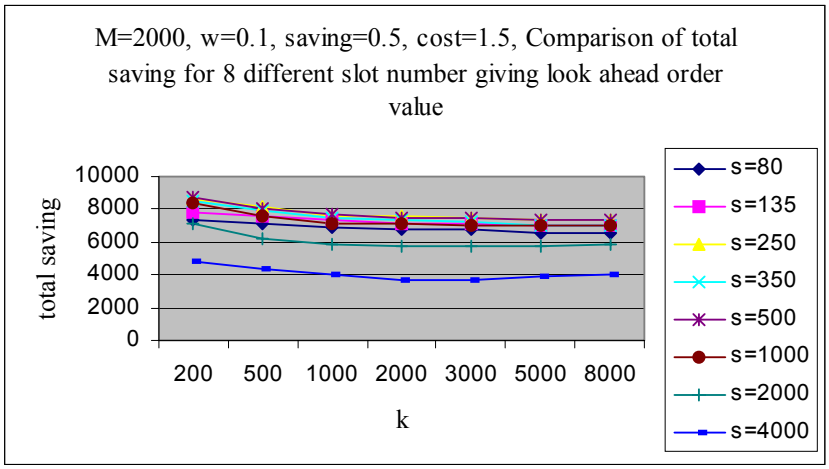

M=3000, w=0.349, saving=0.25, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
3000
2000
1000
0
-1000
-2000
-3000
-4000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

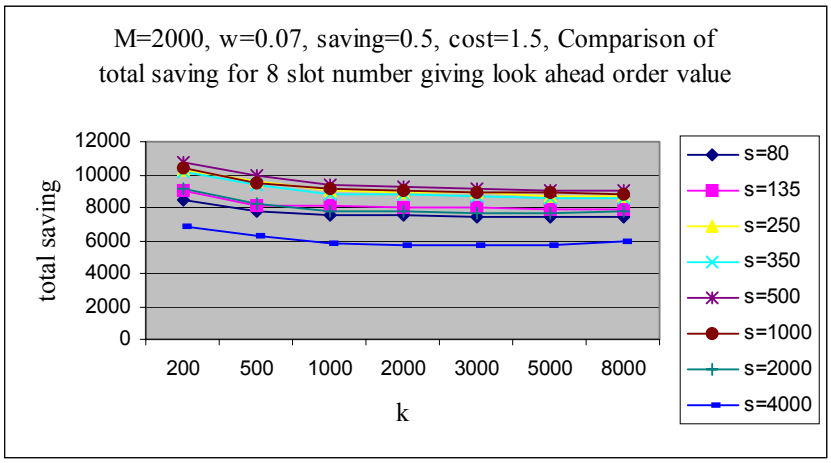

M=3000, w=0.25, saving=0.25, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
4000
2000
0
-2000
-4000
-6000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

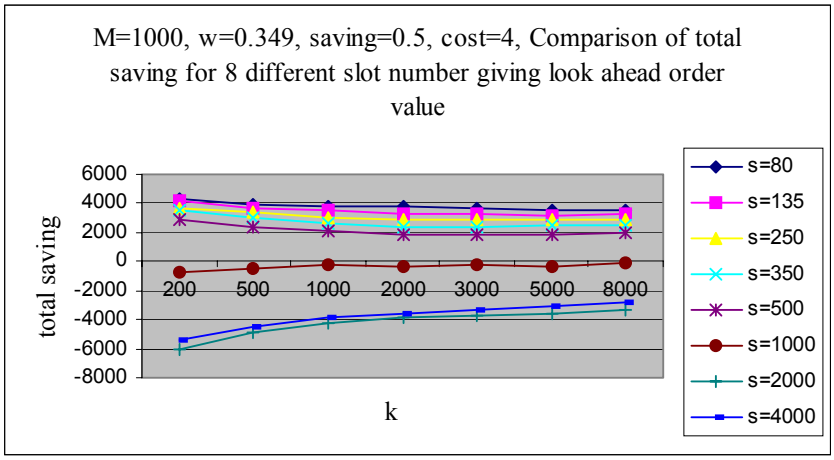

M=3000, w=0.1, saving=0.25, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
4000
2000
0
-2000
-4000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

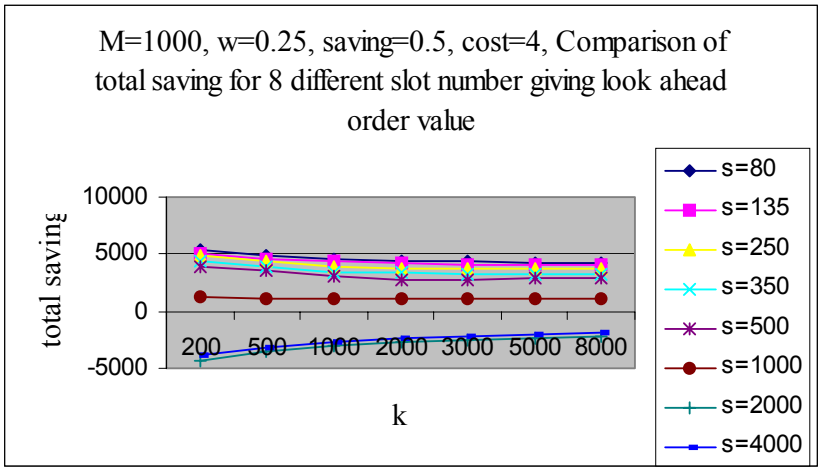

M=2000,w=0.349, saving=0.5, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
4000
2000
0
-2000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

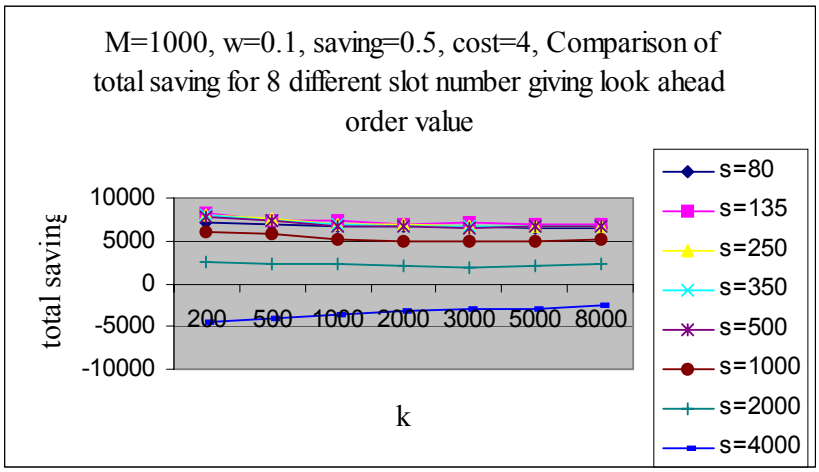

M=2000, w=0.25, saving=0.5, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
4000
2000
0
-2000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

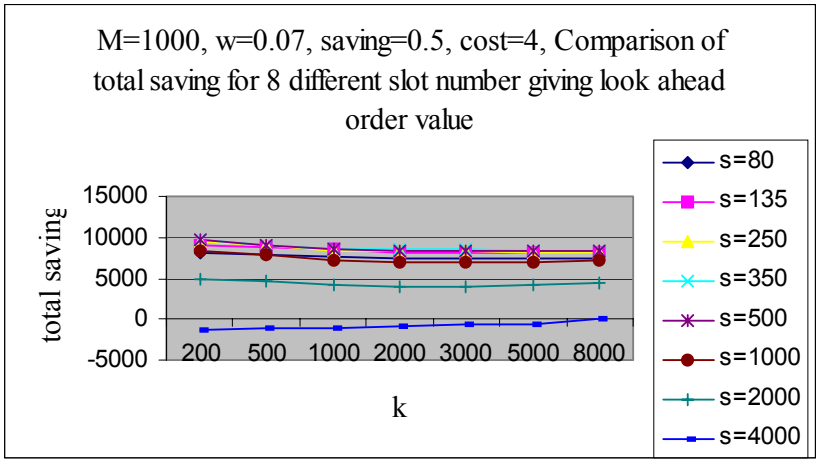

M=2000, w=0.1, saving=0.5, cost=1.5, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
10000
8000
6000
4000
2000
0
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

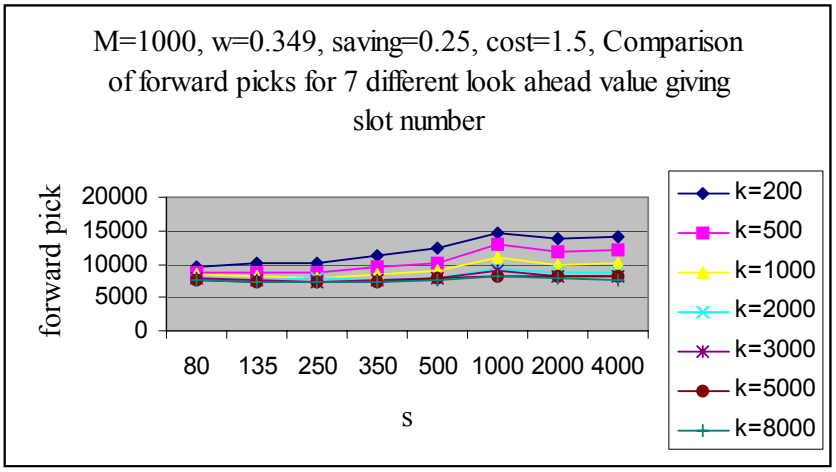

M=2000, w=0.07, saving=0.5, cost=1.5, Comparison of total saving for 8 slot number giving look ahead order value
total saving
12000
10000
8000
6000
4000
2000
0
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

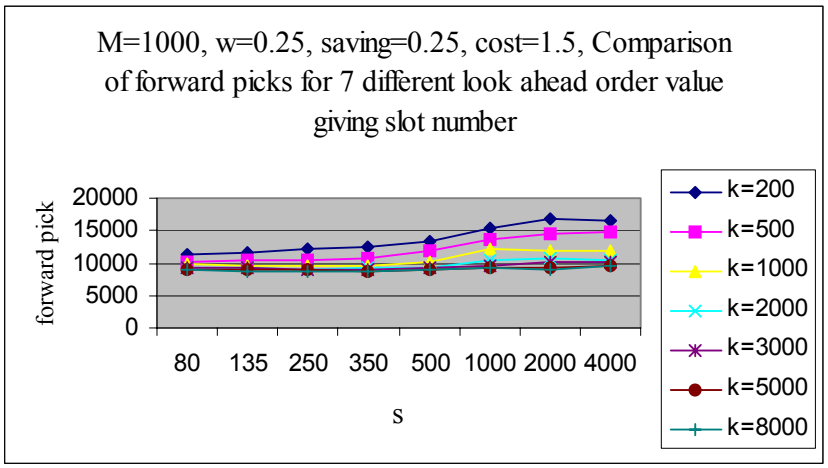

M=1000, w=0.349, saving=0.5, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
6000
4000
2000
0
-2000
-4000
-6000
-8000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

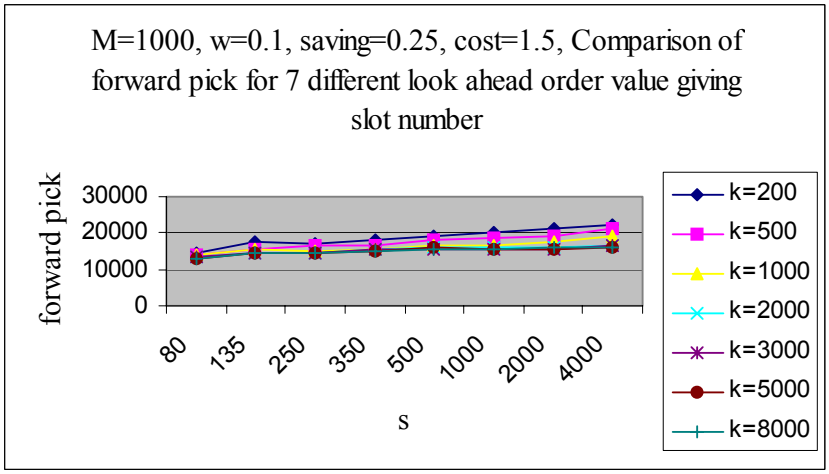

M=1000, w=0.25, saving=0.5, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
10000
5000
0
-5000
200
500
1000
2000
3000
5000
8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

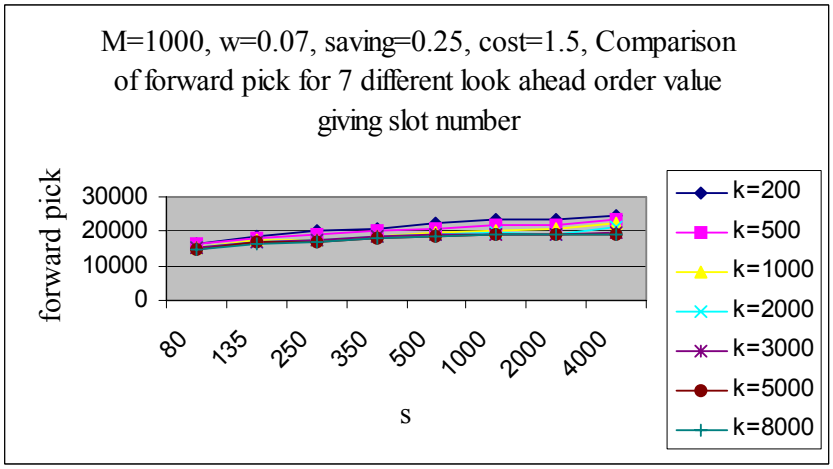

M=1000, w=0.1, saving=0.5, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
10000
5000
0
-5000
-10000
200 500 1000 2000 3000 5000 8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

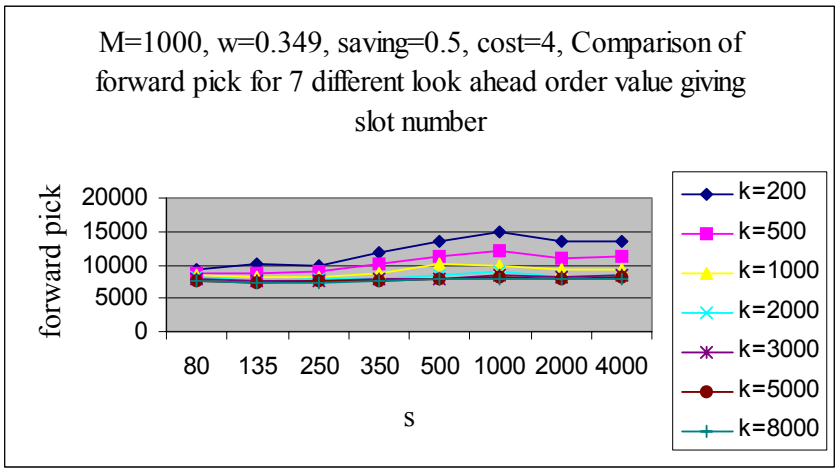

M=1000, w=0.07, saving=0.5, cost=4, Comparison of total saving for 8 different slot number giving look ahead order value
total saving
15000
10000
5000
0
-5000
200 500 1000 2000 3000 5000 8000
k
s=80
s=135
s=250
s=350
s=500
s=1000
s=2000
s=4000

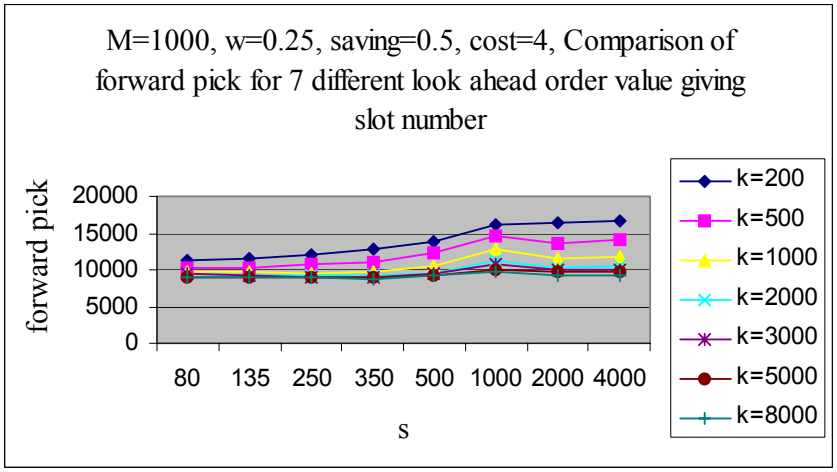

M=1000, w=0.349, saving=0.25, cost=1.5, Comparison of forward picks for 7 different look ahead value giving slot number
forward pick
20000
15000
10000
5000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

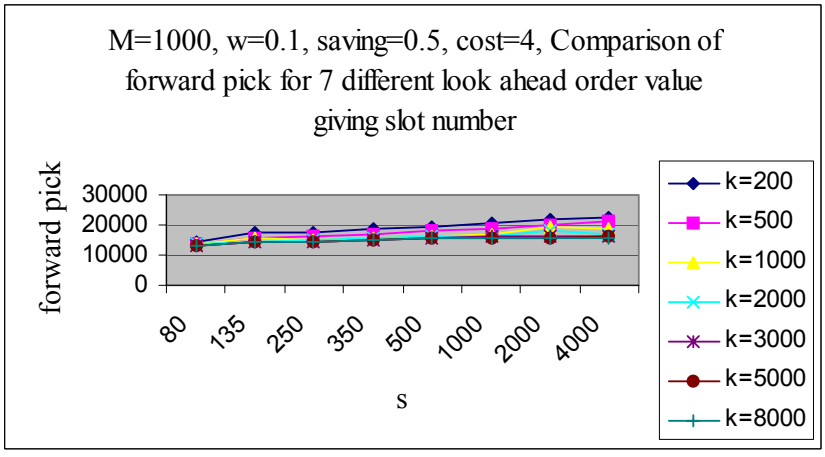

M=1000, w=0.25, saving=0.25, cost=1.5, Comparison of forward picks for 7 different look ahead order value giving slot number
forward pick
20000
15000
10000
5000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

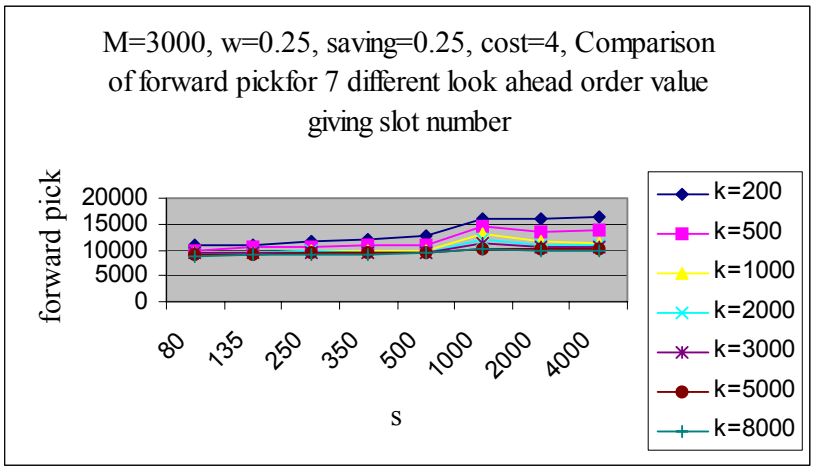

M=1000, w=0.1, saving=0.25, cost=1.5, Comparison of forward pick for 7 different look ahead order value giving slot number
forward pick
30000
20000
10000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

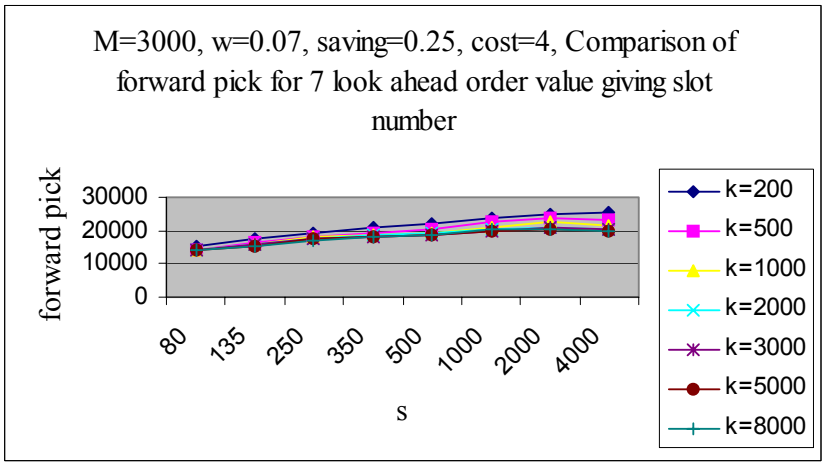

M=1000, w=0.07, saving=0.25, cost=1.5, Comparison of forward pick for 7 different look ahead order value giving slot number
forward pick
30000
20000
10000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

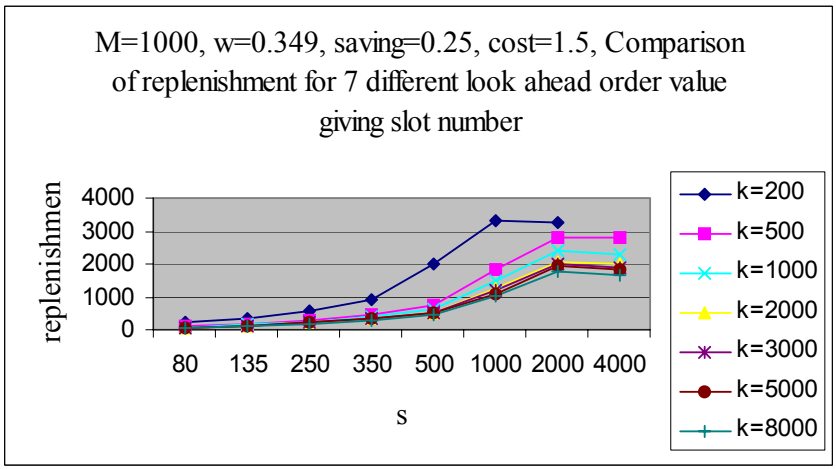

M=1000, w=0.349, saving=0.5, cost=4, Comparison of forward pick for 7 different look ahead order value giving slot number
forward pick
20000
15000
10000
5000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

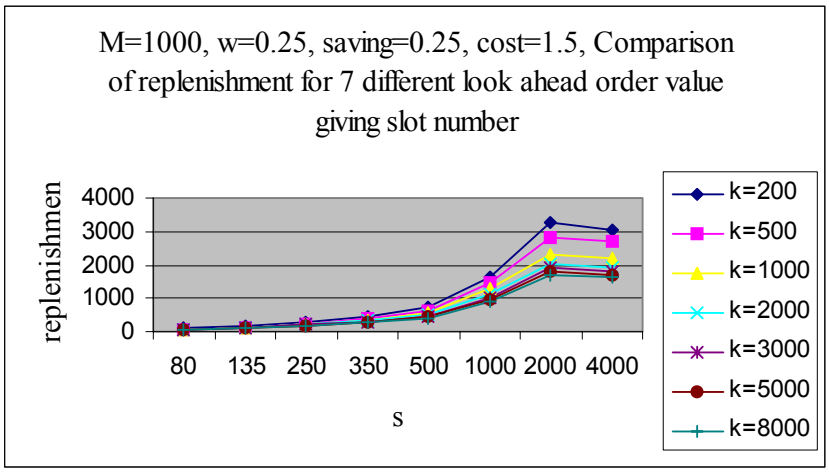

M=1000, w=0.25, saving=0.5, cost=4, Comparison of forward pick for 7 different look ahead order value giving slot number
forward pick
20000
15000
10000
5000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

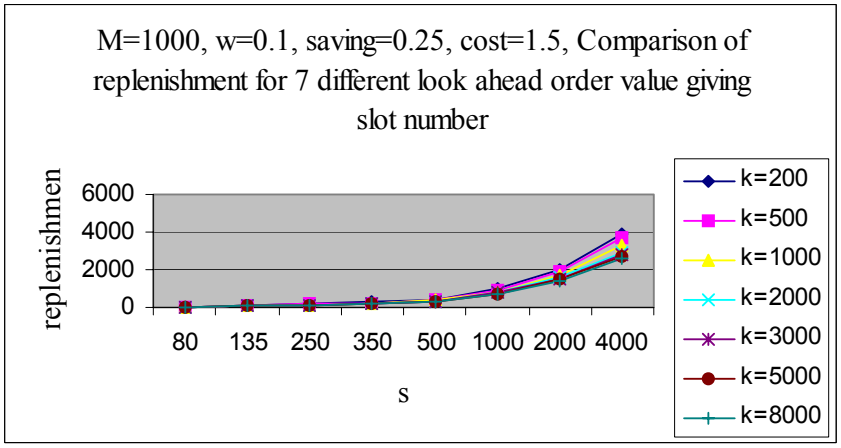

M=1000, w=0.1, saving=0.5, cost=4, Comparison of forward pick for 7 different look ahead order value giving slot number
forward pick
30000
20000
10000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

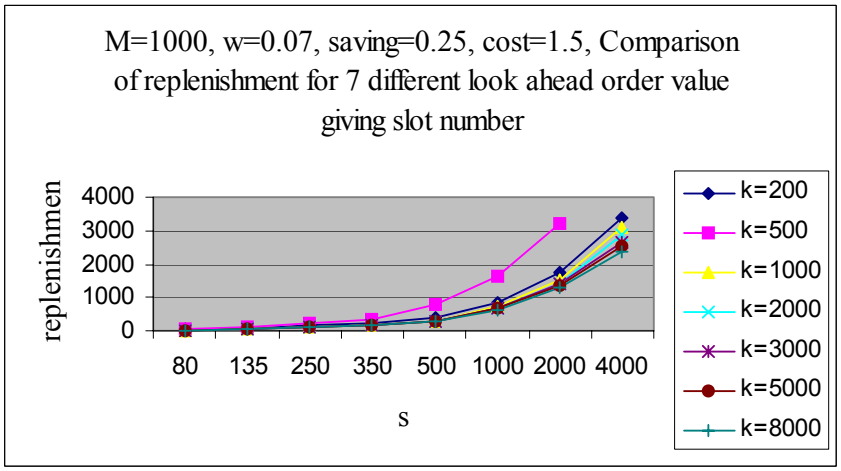

M=3000, w=0.25, saving=0.25, cost=4, Comparison of forward pickfor 7 different look ahead order value giving slot number
forward pick
20000
15000
10000
5000
0
80
135
250
350
500
1000
2000
4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

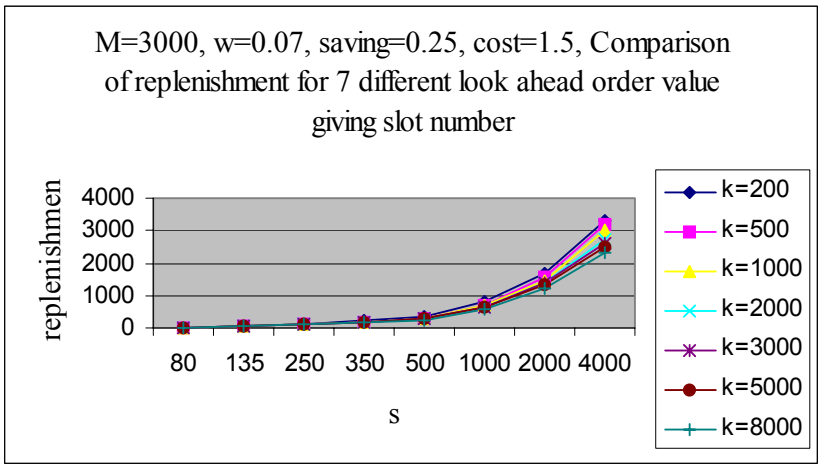

M=3000, w=0.07, saving=0.25, cost=4, Comparison of forward pick for 7 look ahead order value giving slot number
forward pick
30000
20000
10000
0
80
135
250
350
500
1000
2000
4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

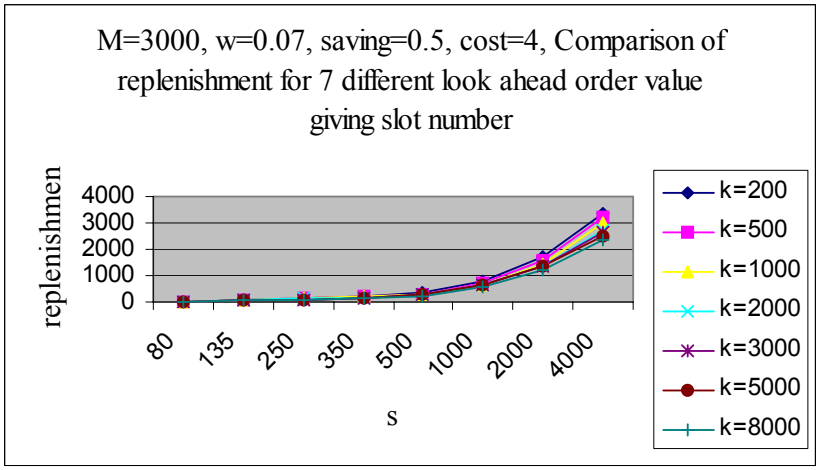

M=1000, w=0.349, saving=0.25, cost=1.5, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
4000
3000
2000
1000
0
80
135
250
350
500
1000
2000
4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

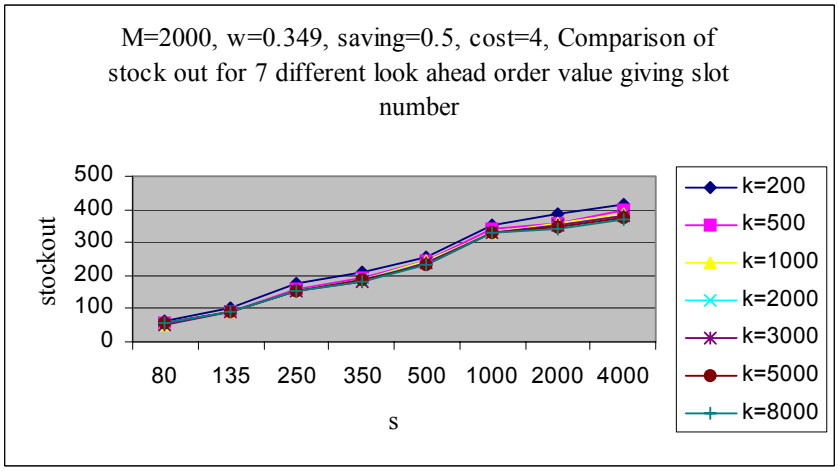

M=1000, w=0.25, saving=0.25, cost=1.5, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

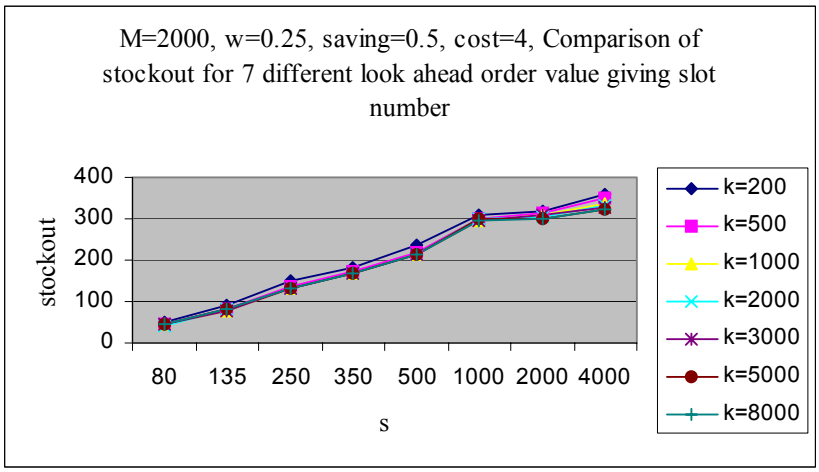

M=1000, w=0.1, saving=0.25, cost=1.5, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
6000
4000
2000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

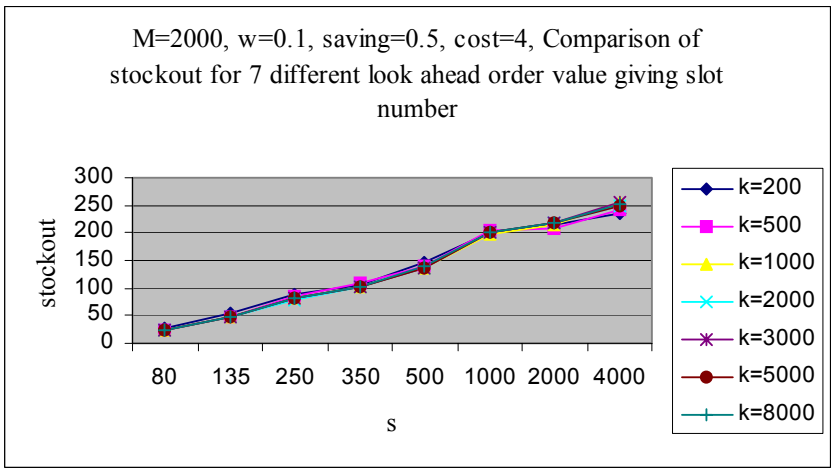

M=1000, w=0.07, saving=0.25, cost=1.5, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

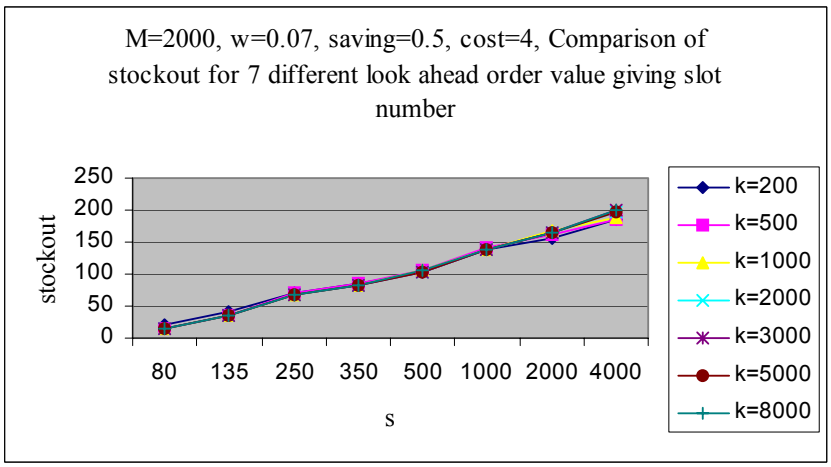

M=3000, w=0.07, saving=0.25, cost=1.5, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000


M=3000, w=0.07, saving=0.5, cost=4, Comparison of replenishment for 7 different look ahead order value giving slot number
replenishmen
4000
3000
2000
1000
0
80
135
250
350
500
1000
2000
4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

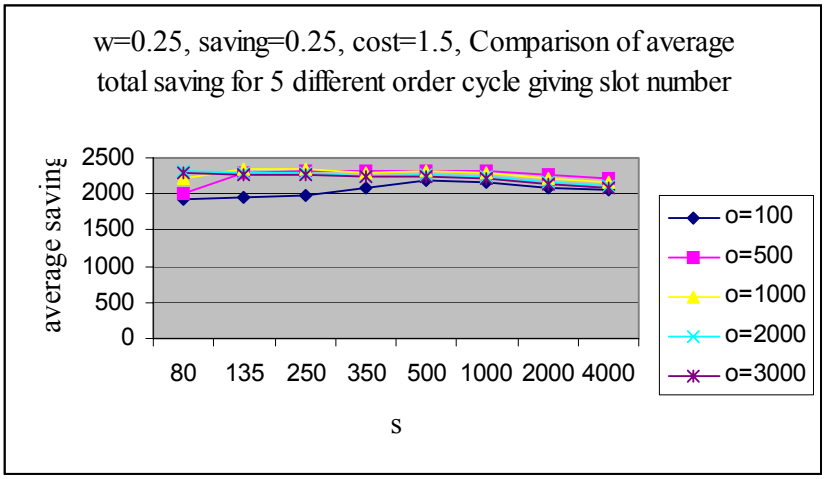

M=2000, w=0.349, saving=0.5, cost=4, Comparison of stock out for 7 different look ahead order value giving slot number
stockout
500
400
300
200
100
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

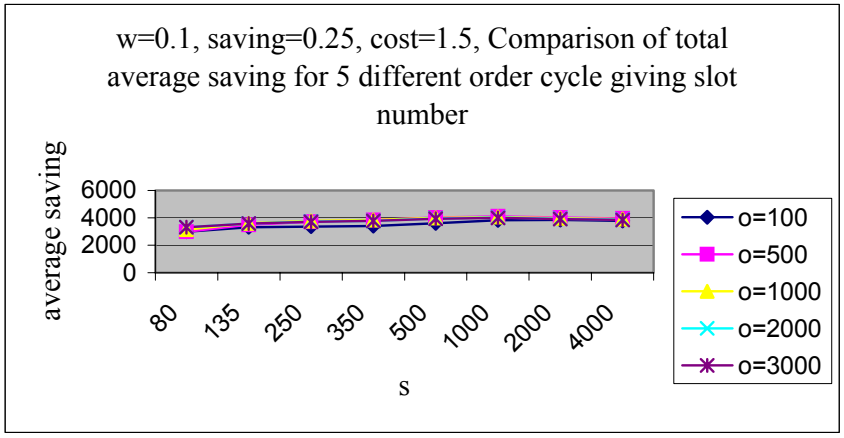

M=2000, w=0.25, saving=0.5, cost=4, Comparison of stockout for 7 different look ahead order value giving slot number
stockout
400
300
200
100
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

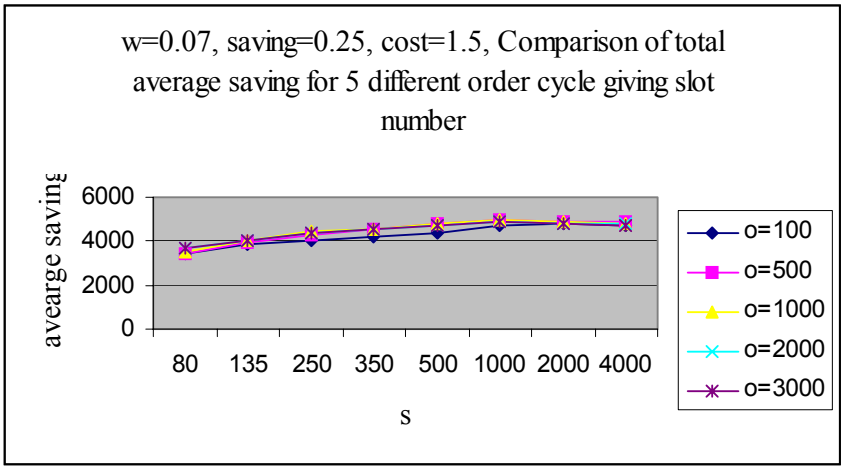

M=2000, w=0.1, saving=0.5, cost=4, Comparison of stockout for 7 different look ahead order value giving slot number
stockout
300
250
200
150
100
50
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

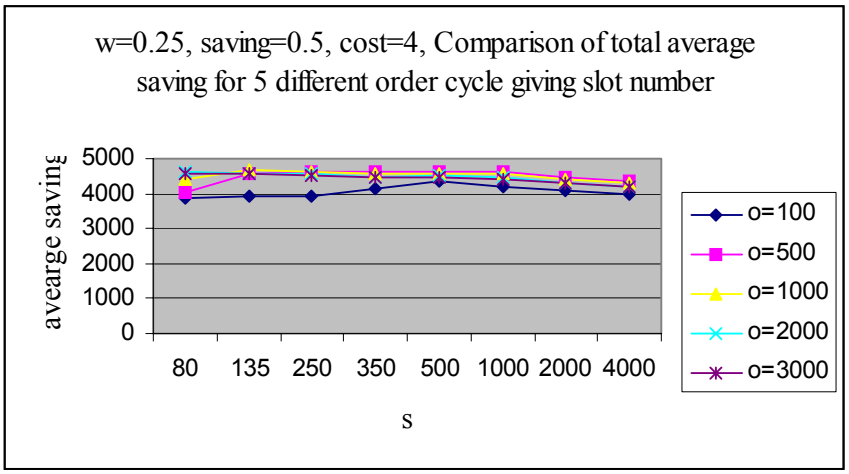

M=2000, w=0.07, saving=0.5, cost=4, Comparison of stockout for 7 different look ahead order value giving slot number
stockout
250
200
150
100
50
0
80 135 250 350 500 1000 2000 4000
s
k=200
k=500
k=1000
k=2000
k=3000
k=5000
k=8000

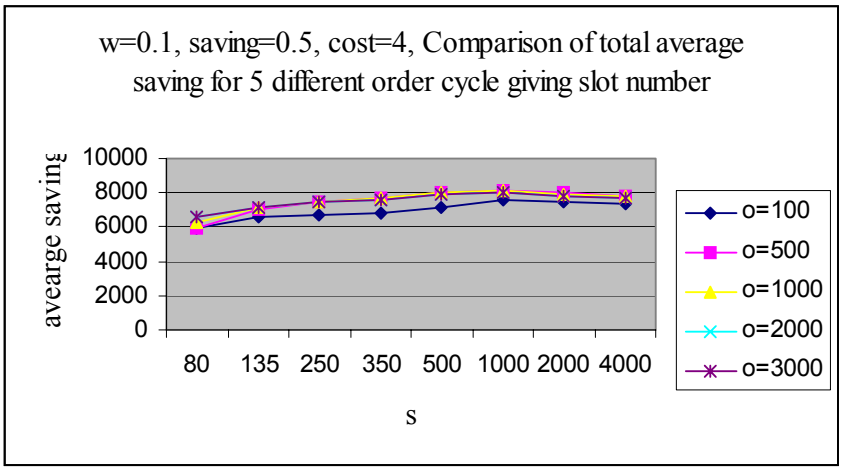
Appendix H : Order-based Dynamic Algorithm Experiments

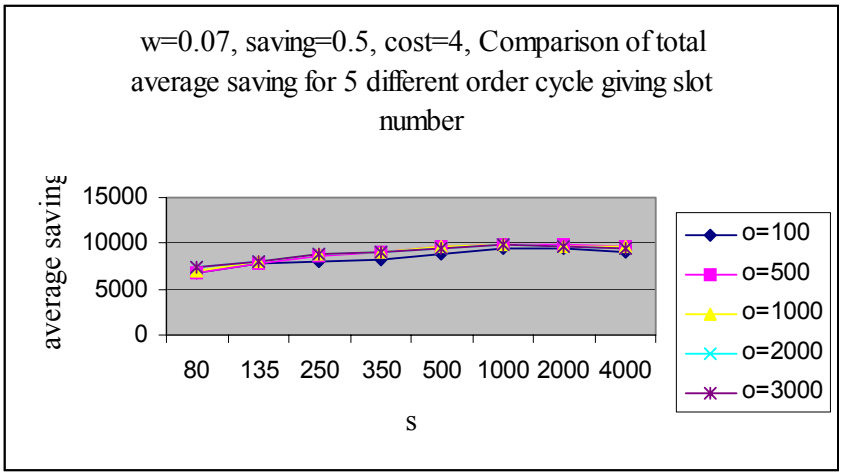


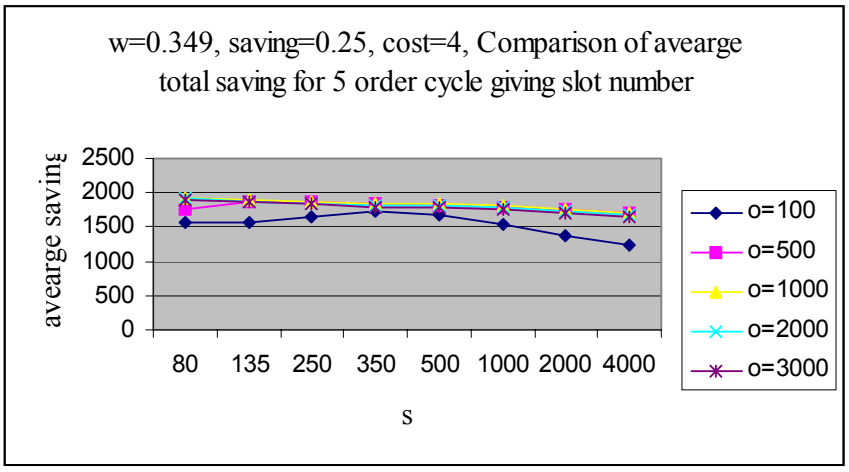


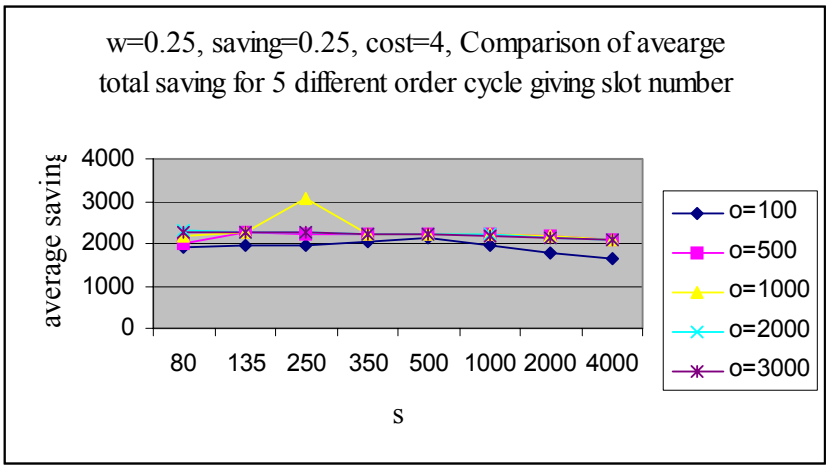


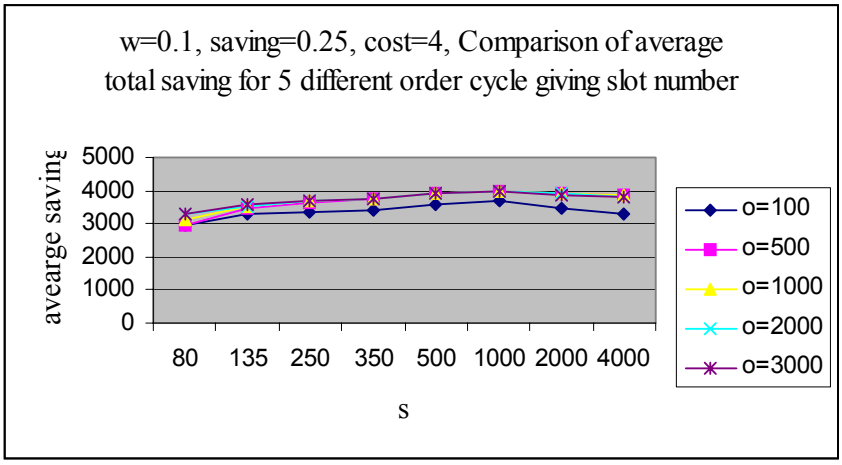

w=0.25, saving=0.5, cost=4, Comparison of total average saving for 5 different order cycle giving slot number
avearge saving
5000
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

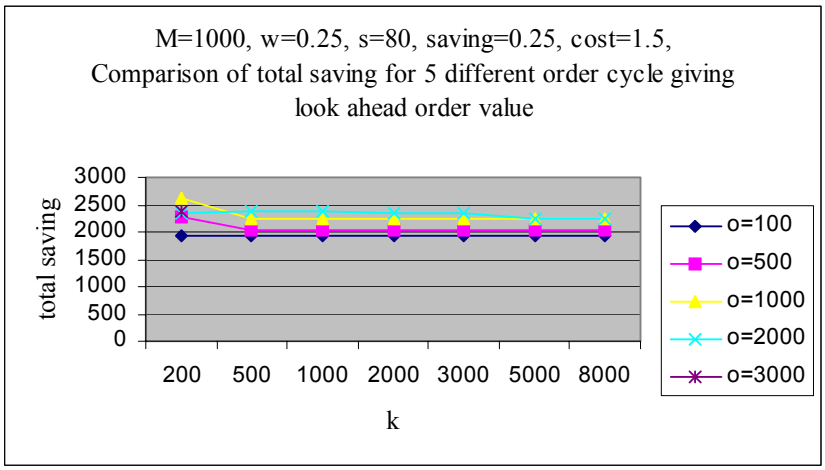

w=0.1, saving=0.5, cost=4, Comparison of total average saving for 5 different order cycle giving slot number
avearge saving
10000
8000
6000
4000
2000
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

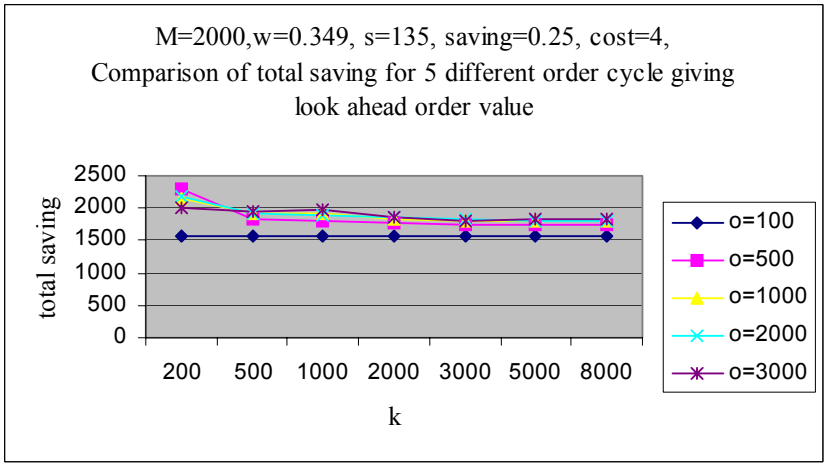

w=0.07, saving=0.5, cost=4, Comparison of total average saving for 5 different order cycle giving slot number
average saving
15000
10000
5000
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

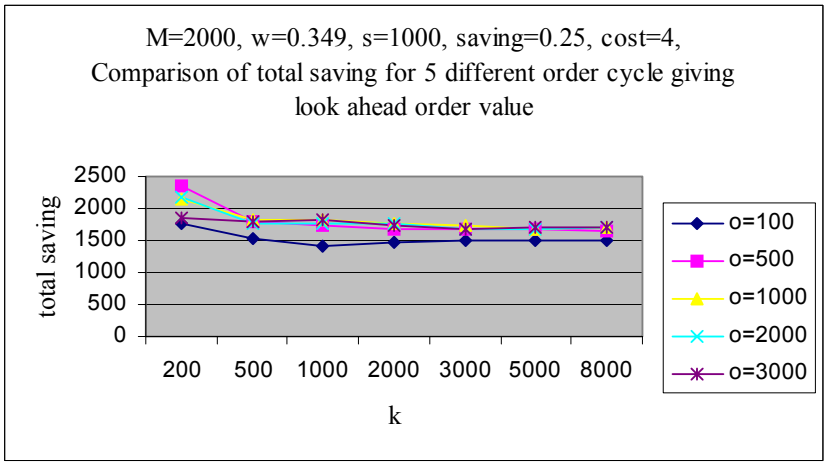

w=0.349, saving=0.25, cost=4, Comparison of avearge total saving for 5 order cycle giving slot number
avearge saving
2500
2000
1500
1000
500
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

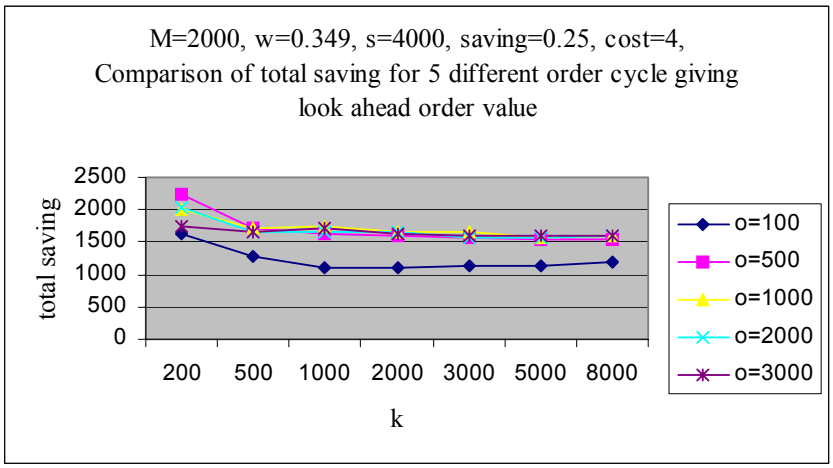

w=0.25, saving=0.25, cost=4, Comparison of avearge total saving for 5 different order cycle giving slot number
average saving
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

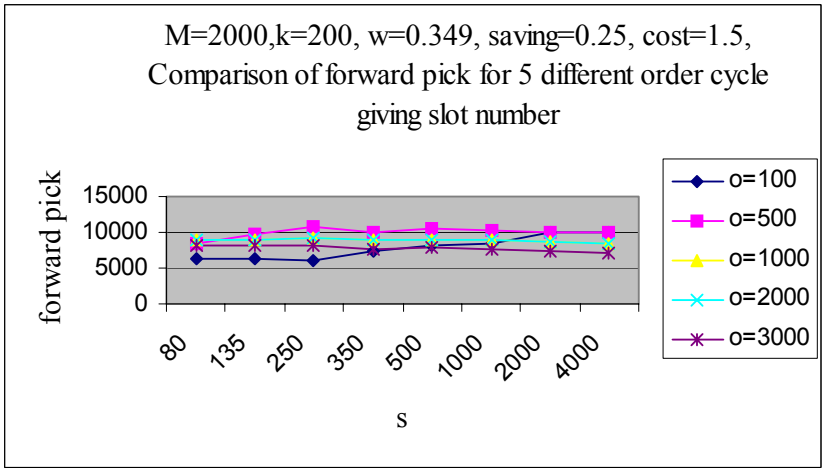

w=0.1, saving=0.25, cost=4, Comparison of average total saving for 5 different order cycle giving slot number
avearge saving
5000
4000
3000
2000
1000
0
80 135 250 350 500 1000 2000 4000
s
o=100
o=500
o=1000
o=2000
o=3000

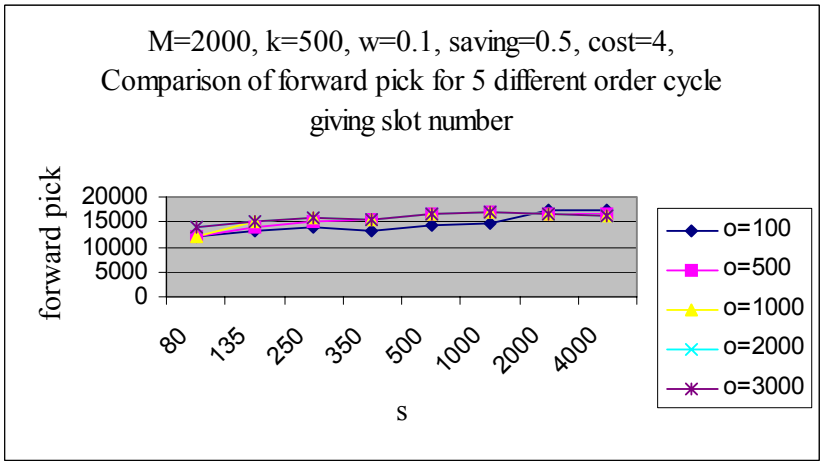

M=1000, w=0.25, s=80, saving=0.25, cost=1.5,
Comparison of total saving for 5 different order cycle giving
look ahead order value
total saving
3000
2500
2000
1500
1000
500
0
200
500
1000
2000
3000
5000
8000
k
o=100
o=500
o=1000
o=2000
o=3000

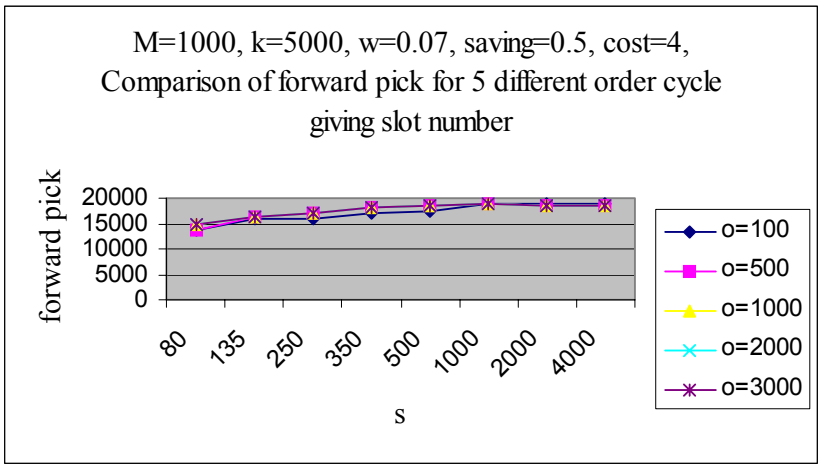

M=2000,w=0.349, s=135, saving=0.25, cost=4,
Comparison of total saving for 5 different order cycle giving
look ahead order value
total saving
2500
2000
1500
1000
500
0
200
500
1000
2000
3000
5000
8000
k
o=100
o=500
o=1000
o=2000
o=3000

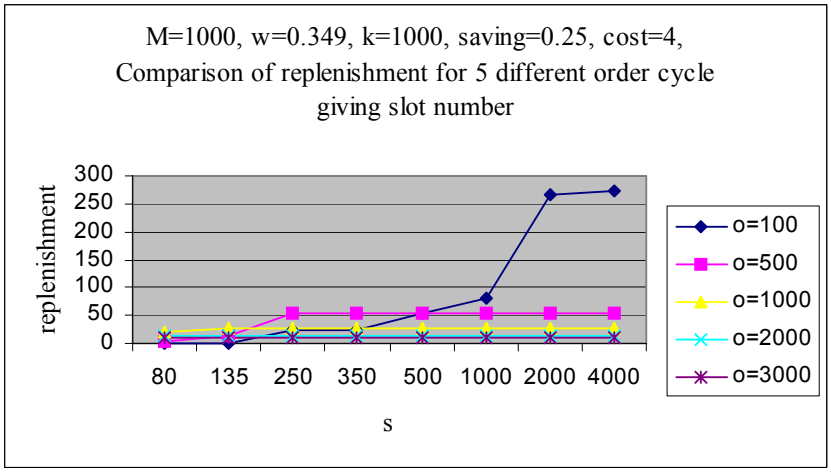

M=2000, w=0.349, s=1000, saving=0.25, cost=4,
Comparison of total saving for 5 different order cycle giving
look ahead order value
total saving
2500
2000
1500
1000
500
0
200
500
1000
2000
3000
5000
8000
k
o=100
o=500
o=1000
o=2000
o=3000

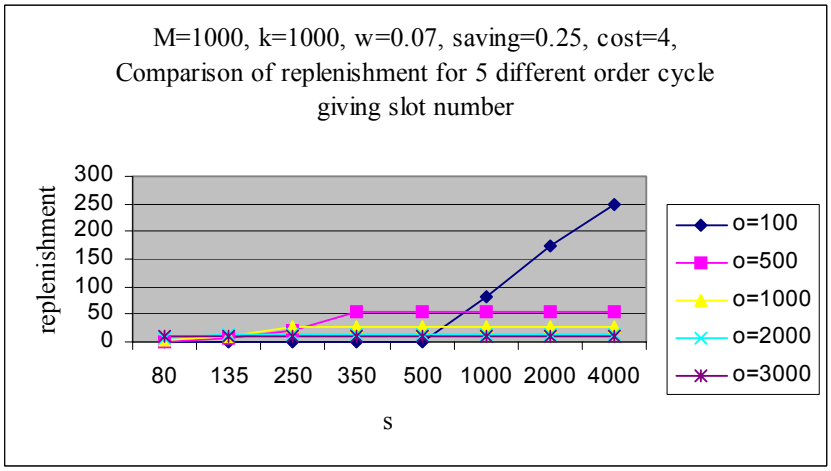

M=2000, w=0.349, s=4000, saving=0.25, cost=4,
Comparison of total saving for 5 different order cycle giving look ahead order value
total saving
2500
2000
1500
1000
500
0
200
500
1000
2000
3000
5000
8000
k
o=100
o=500
o=1000
o=2000
o=3000

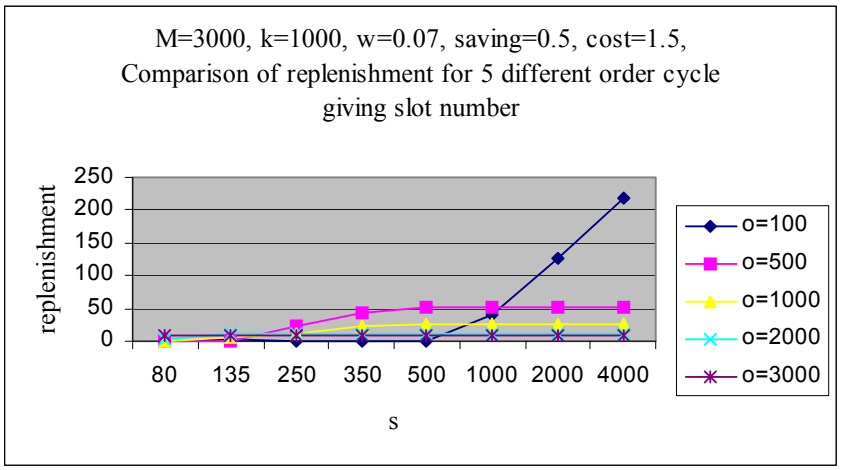

M=2000,k=200, w=0.349, saving=0.25, cost=1.5,
Comparison of forward pick for 5 different order cycle giving slot number
forward pick
15000
10000
5000
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000

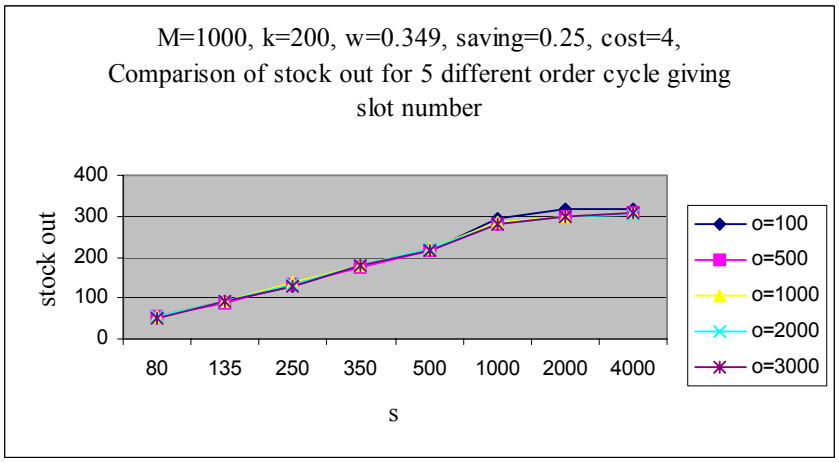

M=2000, k=500, w=0.1, saving=0.5, cost=4,
Comparison of forward pick for 5 different order cycle giving slot number
forward pick
20000
15000
10000
5000
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000

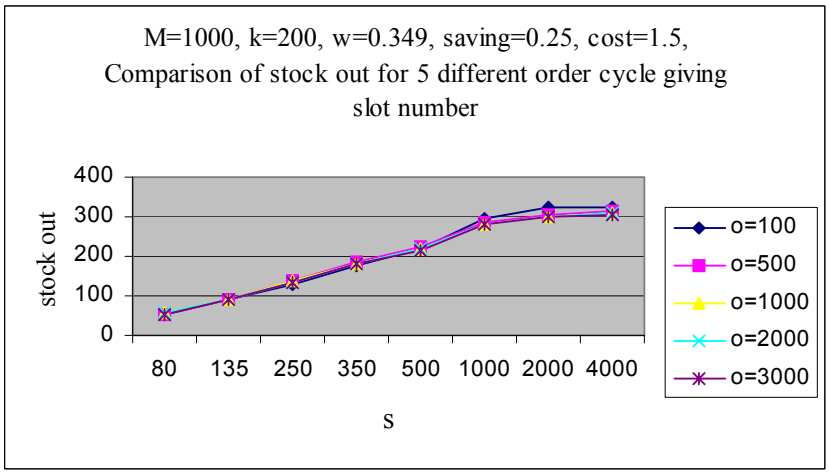

M=1000, k=5000, w=0.07, saving=0.5, cost=4,
Comparison of forward pick for 5 different order cycle giving slot number
forward pick
20000
15000
10000
5000
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000

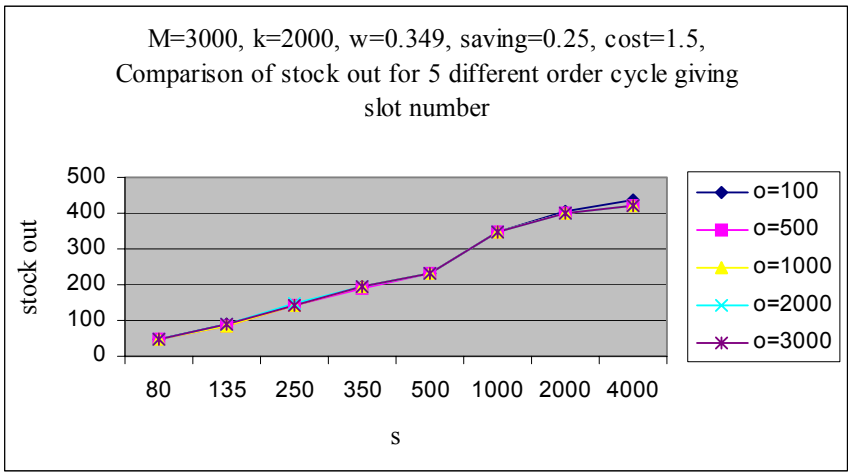

M=1000, w=0.349, k=1000, saving=0.25, cost=4,
Comparison of replenishment for 5 different order cycle giving slot number
replenishment
300
250
200
150
100
50
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000


M=1000, k=1000, w=0.07, saving=0.25, cost=4,
Comparison of replenishment for 5 different order cycle giving slot number
replenishment
300
250
200
150
100
50
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000


M=3000, k=1000, w=0.07, saving=0.5, cost=1.5,
Comparison of replenishment for 5 different order cycle giving slot number
replenishment
250
200
150
100
50
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000


M=1000, k=200, w=0.349, saving=0.25, cost=4,
Comparison of stock out for 5 different order cycle giving slot number
stock out
400
300
200
100
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000


M=1000, k=200, w=0.349, saving=0.25, cost=1.5,
Comparison of stock out for 5 different order cycle giving slot number
stock out
400
300
200
100
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000


M=3000, k=2000, w=0.349, saving=0.25, cost=1.5,
Comparison of stock out for 5 different order cycle giving slot number
stock out
500
400
300
200
100
0
80
135
250
350
500
1000
2000
4000
s
o=100
o=500
o=1000
o=2000
o=3000

10. REFERENCES

1. Bodner, A.D., Govindaraj, T., Blanco, E.E., Goetschalckx, M., McGinnis, F.L., Sharp, P.G.,' Web-based tools in warehouse design', In Proceedings of the 2000 Industrial Engineering Solutions Conference, Cleveland, OH, 2000

2. Bartholdi III, J.J., 'Product layout in a warehouse', ISyE 4101, October 6, 1994

3. Bartholdi III, J.J., Hackman,T.S., Design of a fast pick area, *Chapter 8 in Warehouse & Distribution Science*, Release 0.08, May 22, 1998, revised August 23, 2000

4. Frazelle, H.E., *World-class warehousing and material handling*, McGraw-Hill, 2002

5. Frazelle, H.E., Hackman T.S., Passy, U., Platzman, K.L., The forward-reserve problem, *Chapter 5 in Optimization in industry*, John Wiley & Sons Ltd., 1994

6. Hackman, T.S., Rosenblatt, J.M., 'Allocating items to an automated storage and retrieval system', *IIE Transactions*, 22, pp. 7-14, 1990

7. Hackman, T.S., Platzman, K.L., 'Near-optimal solution of generalized resource allocation problems with large capacities', *Operations Research*, 38, pp. 902-910, 1990

8. Jaikumar, R., Solomon, M.M., 'Dynamic operational policies in an automated warehouse', *IIE Transactions*, 22, pp. 370-376, 1990

9. Malmborg, J.C., Krishnakumar, B., 'Optimal storage assignment policies for multiaddress warehousing systems', *IEEE Transactions On Systems, Man, and Cybernetics,* 19, pp. 197-203, 1989

10. Peters, C., Development of a simulation model representing the forward picking area in a distribution center, Georgia Institute of Technology, 1998

11. Ratliff, D.H., Rosenthal, S.A., 'Order picking in a rectangular warehouse: A solvable case of the traveling salesman problem', *Operations Research*, 31, No:3, pp. 507-521, 1983

12. Sadiq, M., Landers, L.T., Taylor, D.G., 'An assignment algorithm for dynamic picking systems', *IIE Transactions*, 28, pp. 607-616, 1996

13. Sharp, P.G., Amirhosseini, M.M., Schwarz, F., 'New Approaches and results for product storage assignment: Consideration of demand variability, demand correlation, storage compartment size, and degree of order completion', 1998

14. Sharp, P.G., Warehouse Management, *Chapter 81 in Handbook of Industrial Engineering,* Gavriel Salvendy, Ed., John Wiley & Sons, Inc., New York, 2000

15. Tompkins, A.J., White, A.J., Bozer, A.Y., Frazelle, H.E., Tanchoco, A.M.J., Trevino, J., *Facilities planning*, John Wiley & Sons, 1996

16. Van den Berg, P.J., Sharp, P.G., Gademann, M.R.J.A., Pochet, Y., 'Forward-

reserve allocation in a warehouse with unit load replenishments', *European Journal of Operational Research*, 111, pp. 98-113, 1998

17. Van den Berg, P.J., Zijm, M.H.W., 'Models for warehouse management: Classification and examples', *Int. J. Production Economics*, 59, pp. 519-528, 1999

18. Yoon, S.C., Sharp, P.G., 'A structured procedure for analysis and design of order pick systems', *IIE Transactions*, 28, pp. 379-389, 1996

10. REFERENCES

1. Bodner, A.D., Govindaraj, T., Blanco, E.E., Goetschalckx, M., McGinnis, F.L., Sharp, P.G.,' Web-based tools in warehouse design', In Proceedings of the 2000 Industrial Engineering Solutions Conference, Cleveland, OH, 2000

2. Bartholdi III, J.J., 'Product layout in a warehouse', ISyE 4101, October 6, 1994

3. Bartholdi III, J.J., Hackman,T.S., Design of a fast pick area, *Chapter 8 in Warehouse & Distribution Science*, Release 0.08, May 22, 1998, revised August 23, 2000

4. Frazelle, H.E., *World-class warehousing and material handling*, McGraw-Hill, 2002

5. Frazelle, H.E., Hackman T.S., Passy, U., Platzman, K.L., The forward-reserve problem, *Chapter 5 in Optimization in industry*, John Wiley & Sons Ltd., 1994

6. Hackman, T.S., Rosenblatt, J.M., 'Allocating items to an automated storage and retrieval system', *IIE Transactions*, 22, pp. 7-14, 1990

7. Hackman, T.S., Platzman, K.L., 'Near-optimal solution of generalized resource allocation problems with large capacities', *Operations Research*, 38, pp. 902-910, 1990

8. Jaikumar, R., Solomon, M.M., 'Dynamic operational policies in an automated warehouse', *IIE Transactions*, 22, pp. 370-376, 1990

9. Malmborg, J.C., Krishnakumar, B., 'Optimal storage assignment policies for multiaddress warehousing systems', *IEEE Transactions On Systems, Man, and Cybernetics,* 19, pp. 197-203, 1989

10. Peters, C., Development of a simulation model representing the forward picking area in a distribution center, Georgia Institute of Technology, 1998

11. Ratliff, D.H., Rosenthal, S.A., 'Order picking in a rectangular warehouse: A solvable case of the traveling salesman problem', *Operations Research*, 31, No:3, pp. 507-521, 1983

12. Sadiq, M., Landers, L.T., Taylor, D.G., 'An assignment algorithm for dynamic picking systems', *IIE Transactions*, 28, pp. 607-616, 1996

13. Sharp, P.G., Amirhosseini, M.M., Schwarz, F., 'New Approaches and results for product storage assignment: Consideration of demand variability, demand correlation, storage compartment size, and degree of order completion', 1998

14. Sharp, P.G., Warehouse Management, *Chapter 81 in Handbook of Industrial Engineering,* Gavriel Salvendy, Ed., John Wiley & Sons, Inc., New York, 2000

15. Tompkins, A.J., White, A.J., Bozer, A.Y., Frazelle, H.E., Tanchoco, A.M.J., Trevino, J., *Facilities planning*, John Wiley & Sons, 1996

16. Van den Berg, P.J., Sharp, P.G., Gademann, M.R.J.A., Pochet, Y., 'Forward-

reserve allocation in a warehouse with unit load replenishments', *European Journal of Operational Research*, 111, pp. 98-113, 1998

17. Van den Berg, P.J., Zijm, M.H.W., 'Models for warehouse management: Classification and examples', *Int. J. Production Economics*, 59, pp. 519-528, 1999

18. Yoon, S.C., Sharp, P.G., 'A structured procedure for analysis and design of order pick systems', *IIE Transactions*, 28, pp. 379-389, 1996