A MODEL FOR DETERMINING THE NUMBER OF STORAGE BLOCKS AND BLOCK LENGTHS IN A RECTANGULAR WAREHOUSE

by
BİLGE KÜÇÜK

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
June 2003

A MODEL FOR DETERMINING THE NUMBER OF

STORAGE BLOCKS AND BLOCK LENGTHS

IN A RECTANGULAR WAREHOUSE

APPROVED BY:

Dr. Gürdal Ertek …….

(Thesis Advisor)

Dr. Kemal Kılıç …….

Dr. Yücel Saygın …….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Gürdal Ertek, for his patience, encouragement, help for learning Java and financial support throughout this study. I would like to thank him especially for his invaluable guidance while overcoming many challenges along this research.

I would like to thank to my entire friend and I am especially grateful to Özkan Öztürk, Evren Burcu Kıvanç and Şilan Hun, with whom I have been sharing my life in Sabanci University, for their emotional support, friendship, caring and patience along hard times. I will never forget the times we spent with Şilan as flatmates.

I wish to thank my entire extended family for providing a loving environment for me. I am grateful due to their encouragement, love, and generosity.

Finally to my love Serkan Incel. His compassion, tolerance, and patience toward me have been so helpful and generous along this study. I could not overcome difficulties without his encouragement and love. I need to express my special gratitude for everything he has been doing for me.

ABSTRACT

This thesis study focuses on the savings that can be attained by adding unequally spaced cross aisles in a rectangular warehouse and the best locations of these cross aisles.

Earlier research suggests that adding cross aisles perpendicular to main aisles can bring significant savings with respect to order picking travel distance. In the related literature cross aisles are distributed between storage blocks that are equal in length. In this thesis, the locations of cross aisles will be investigated in detail and evaluated in terms of travel distance and storage space. Storage block lengths between cross aisles are allowed to be different in length. Based on our experiment results, these unequally spaced cross aisle configurations enable more saving on order picking travel distance than equally spaced cross aisles. We have also investigated the issue of possible space savings by using less number of unequally spaced cross aisles compared to equally spaced cross aisles. Our research suggests the same travel distance saving due to equally spaced cross aisles can be achieved with less number of cross aisles (less number of storage blocks) distributed between block lengths (cross aisle spacing) that are unequal in length.

Additionally, an interesting pattern in terms of the lengths of storage blocks between unequally spaced cross aisles is observed. In the configuration of storage blocks that provide maximum travel distance saving, the length of the block in the middle gets wider as order size increases. This pattern is observed for all of the warehouse types investigated in this thesis.

Since warehouse design is a strategic problem concerning long-term investment, the results provided within this thesis provide insights in order to decrease the investment costs or increase the efficiency of warehouse operations.

ÖZET

Bu tez dikdörtgen bir depo alanına ara koridorlar ekleyerek elde edilecek kazançları ve bu ara koridorların en iyi yerleşimini konu almaktadır.

Geçmiş çalışmalar ana stok koridorlarını dik kesen ara koridorlar koymanın sipariş toplamak için alınan yol uzunluğunda önemli kazançlar sağladığı göstermiştir. İlgili literatürde ara koridorlar eşit uzunluktaki stok blokları arasına dağıtılmıştır. Bu tezde, ara koridorların yerleşimi, yürünen yol uzunluğunda ve kullanılan stok alanında meydana getirdikleri kazanç bakımından incelenmiştir. Ara koridorlar arasındaki stok blok uzunluklarının birbirinden farklı olduğu durumlar detaylıca incelenmiştir. Elde edilen deneysel sonuçlardan yola çıkarak eşit olmayan aralıklarla dağılan ara koridor kombinasyonlarının eşit aralıklarla dağılan koridorlardan toplam sipariş toplama yürüme mesafesinde daha iyi kazançlar sağladığı sonucuna varılmıştır. Eşit aralıklı ara koridorlara göre daha az sayıda eşit olmayan aralıklı ara koridor kullanarak sağlanabilecek stok alanı kazançları da incelenmiştir. Bizim çalışmamız eşit aralıklı ara koridorlar sayesinde elde edilen yürüyüş mesafesi kazancının daha az sayıda ara koridorun eşit olmayan stok blokları arasına yerleştirilmesiyle elde edilebileceğini göstermiştir.

Buna ek olarak, eşit olmayan aralıklı ara koridorların arasındaki stok bloklarının uzunluklarına ilişkin ilginç bir biçim gözlenmiştir. Yürüyüş mesafesinde en çok kazancı sağlayan stok blokları konfigürasyonunda, orta kısımdaki bloğun uzunluğunun sipariş büyüklüğü arttıkça arttığı gözlenmiştir. Bu gözlem bu tezde incelenen tüm depo tiplerinde tekrarlanmıştır.

Depo tasarımı uzun vadeli yatırımları içeren stratejik bir konu olduğu için bu tezde sunulan sonuçlar yatırım maliyetlerini düşürmek ya da depo operasyonlarının etkinliğini arttırmak için ip uçları sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

A MODEL FOR DETERMINING THE NUMBER OF STORAGE BLOCKS AND BLOCK LENGTHS IN A RECTANGULAR WAREHOUSE

by
BİLGE KÜÇÜK

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
June 2003

A MODEL FOR DETERMINING THE NUMBER OF

STORAGE BLOCKS AND BLOCK LENGTHS

IN A RECTANGULAR WAREHOUSE

APPROVED BY:

Dr. Gürdal Ertek …….

(Thesis Advisor)

Dr. Kemal Kılıç …….

Dr. Yücel Saygın …….

DATE OF APPROVAL: …….

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor, Dr. Gürdal Ertek, for his patience, encouragement, help for learning Java and financial support throughout this study. I would like to thank him especially for his invaluable guidance while overcoming many challenges along this research.

I would like to thank to my entire friend and I am especially grateful to Özkan Öztürk, Evren Burcu Kıvanç and Şilan Hun, with whom I have been sharing my life in Sabanci University, for their emotional support, friendship, caring and patience along hard times. I will never forget the times we spent with Şilan as flatmates.

I wish to thank my entire extended family for providing a loving environment for me. I am grateful due to their encouragement, love, and generosity.

Finally to my love Serkan Incel. His compassion, tolerance, and patience toward me have been so helpful and generous along this study. I could not overcome difficulties without his encouragement and love. I need to express my special gratitude for everything he has been doing for me.

ABSTRACT

This thesis study focuses on the savings that can be attained by adding unequally spaced cross aisles in a rectangular warehouse and the best locations of these cross aisles.

Earlier research suggests that adding cross aisles perpendicular to main aisles can bring significant savings with respect to order picking travel distance. In the related literature cross aisles are distributed between storage blocks that are equal in length. In this thesis, the locations of cross aisles will be investigated in detail and evaluated in terms of travel distance and storage space. Storage block lengths between cross aisles are allowed to be different in length. Based on our experiment results, these unequally spaced cross aisle configurations enable more saving on order picking travel distance than equally spaced cross aisles. We have also investigated the issue of possible space savings by using less number of unequally spaced cross aisles compared to equally spaced cross aisles. Our research suggests the same travel distance saving due to equally spaced cross aisles can be achieved with less number of cross aisles (less number of storage blocks) distributed between block lengths (cross aisle spacing) that are unequal in length.

Additionally, an interesting pattern in terms of the lengths of storage blocks between unequally spaced cross aisles is observed. In the configuration of storage blocks that provide maximum travel distance saving, the length of the block in the middle gets wider as order size increases. This pattern is observed for all of the warehouse types investigated in this thesis.

Since warehouse design is a strategic problem concerning long-term investment, the results provided within this thesis provide insights in order to decrease the investment costs or increase the efficiency of warehouse operations.

ÖZET

Bu tez dikdörtgen bir depo alanına ara koridorlar ekleyerek elde edilecek kazançları ve bu ara koridorların en iyi yerleşimini konu almaktadır.

Geçmiş çalışmalar ana stok koridorlarını dik kesen ara koridorlar koymanın sipariş toplamak için alınan yol uzunluğunda önemli kazançlar sağladığı göstermiştir. İlgili literatürde ara koridorlar eşit uzunluktaki stok blokları arasına dağıtılmıştır. Bu tezde, ara koridorların yerleşimi, yürünen yol uzunluğunda ve kullanılan stok alanında meydana getirdikleri kazanç bakımından incelenmiştir. Ara koridorlar arasındaki stok blok uzunluklarının birbirinden farklı olduğu durumlar detaylıca incelenmiştir. Elde edilen deneysel sonuçlardan yola çıkarak eşit olmayan aralıklarla dağılan ara koridor kombinasyonlarının eşit aralıklarla dağılan koridorlardan toplam sipariş toplama yürüme mesafesinde daha iyi kazançlar sağladığı sonucuna varılmıştır. Eşit aralıklı ara koridorlara göre daha az sayıda eşit olmayan aralıklı ara koridor kullanarak sağlanabilecek stok alanı kazançları da incelenmiştir. Bizim çalışmamız eşit aralıklı ara koridorlar sayesinde elde edilen yürüyüş mesafesi kazancının daha az sayıda ara koridorun eşit olmayan stok blokları arasına yerleştirilmesiyle elde edilebileceğini göstermiştir.

Buna ek olarak, eşit olmayan aralıklı ara koridorların arasındaki stok bloklarının uzunluklarına ilişkin ilginç bir biçim gözlenmiştir. Yürüyüş mesafesinde en çok kazancı sağlayan stok blokları konfigürasyonunda, orta kısımdaki bloğun uzunluğunun sipariş büyüklüğü arttıkça arttığı gözlenmiştir. Bu gözlem bu tezde incelenen tüm depo tiplerinde tekrarlanmıştır.

Depo tasarımı uzun vadeli yatırımları içeren stratejik bir konu olduğu için bu tezde sunulan sonuçlar yatırım maliyetlerini düşürmek ya da depo operasyonlarının etkinliğini arttırmak için ip uçları sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Scope of the Study

This thesis focuses on the following fundamental question: “What is the best configuration of the storage blocks in a rectangular warehouse with cross aisles?” The rectangular warehouse that we consider is illustrated in Figure 1.1.

The characteristics of the rectangular warehouse that we consider are assumed as follows:

- There are parallel main aisles, where products are stored on both sides of main aisles
- All stocks of a particular part are stored in a single location
- Order pickers can traverse the aisles in both directions and change directions within the main aisles
- Pickers travel to parts
- The main aisles are narrow enough to pick from both sides of the aisle without changing position
- There are two natural cross aisles in the warehouse, at the head and rear of the warehouse
- Cross aisles do not contain storage locations, but can be used to change main aisle

- Each order includes a number of items to be picked, which are generally located in various main aisles
- It is considered that the items of an order are collected in a single tour
- When an order is received an order picker is sent from the dispatching area and collected products are gathered in the consolidation area

In this thesis, "number of cross aisles" refers to the number of interior cross aisles, which divide the main aisles, and are between the head and rear natural cross aisles. We assume that picking tours start and end at the southeast and southwest corners of the warehouse, respectively. Even though some research assumes that order picking ends at the starting point (de Koster and van der Poort (1998), Roodbergen and de Koster (2001)), this does not make a great change in travel distance: Petersen (1997) notes that this change results in at most 1% deviation in travel distance.

Chapter 2 presents a review of the related literature.

Chapter 3 the modified shortest path model is given. This model is the extension of the dynamic algorithm introduced by Vaughan and Petersen (1999). The modification reflects the structure of unequally spaced cross aisles.

Chapter 4 explains the layout grid search algorithms. These algorithms are generated in order to investigate as many as possible different warehouse layouts for various warehouse types and order sets.

Chapter 5 includes the experimental design that is conducted for investigation of impacts of unequally spaced cross aisles for various warehouse types and order sets.

Chapter 6 presents the experimental results. Unequally spaced cross aisles are compared with equally spaced cross aisles in terms of the order picking travel distance. It is observed that unequally spaced cross aisles provide more travel distance saving than equally cross aisles do. Moreover, since less number unequally spaced cross aisles are as efficient as more number of equally spaced cross aisles, unequally spaced cross

aisles enable storage space saving, too. Additionally, an interesting pattern of length of storage space is observed as order sizes increase.

Summarizing the results achieved, a conclusion of the study is provided in Chapter 7.

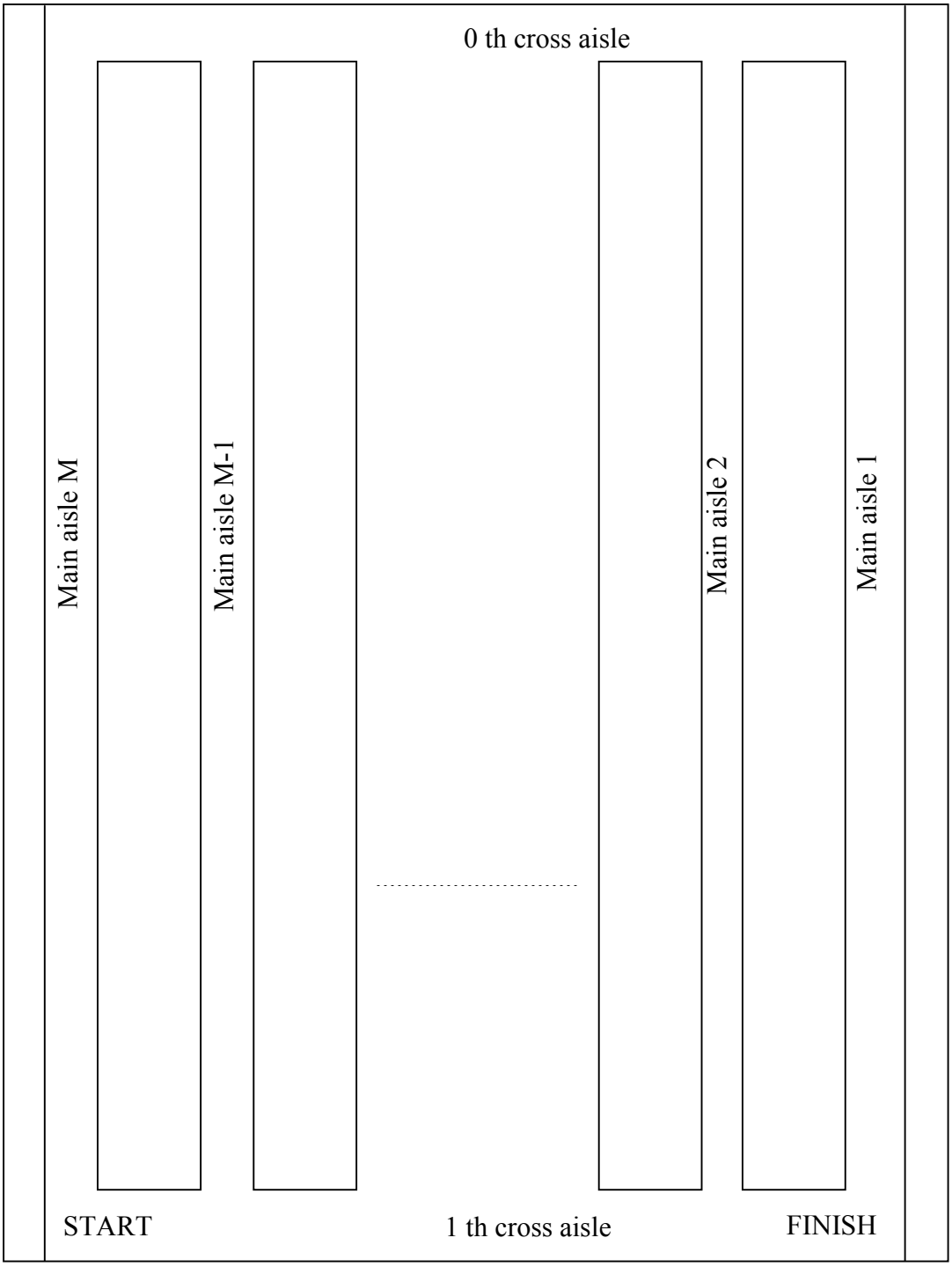


Figure 1.1. A rectangular warehouse

2. LITERATURE REVIEW

2.1. The Supply Chains

There is an increasing trend towards supply chains flowing smoothly. Increasing product varieties, decreasing product life cycles and response times force companies to have great emphasis on the logistic operations. Sharp competition circumstances of the trade markets influence companies to decrease their costs as much as possible, and efficient logistic operations are crucial for cost reduction.

Concerning logistic operations as networks, warehouses constitute the nodes of this network and their efficiency can have a tremendous impact on the whole network. Frazelle (2002) explains the trends in warehousing: "Not too long ago, effective warehousing was a relatively straightforward progression of receiving, storing, and shipping. But in today's age of e-commerce, supply chain integration, globalisation, and just-in-time methodology, warehousing has become more complex than at any time in the past, not to mention more costly".

As depicted in Figure 2.1, warehousing is a significant cost component in supply chain activities. Within the scope of this thesis more attention will be spent on warehouse processes.

(excluding physical supply)

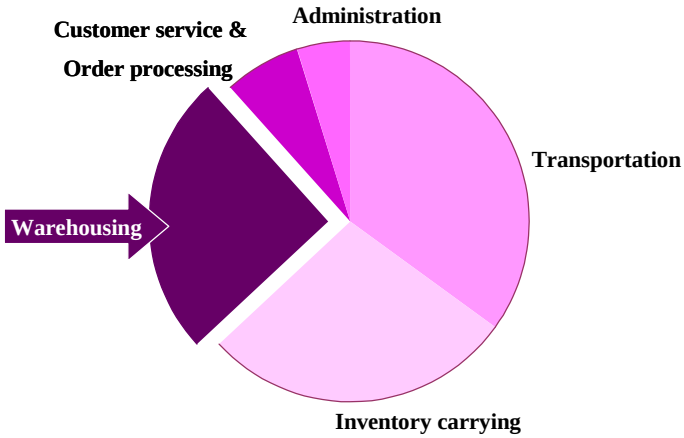


Figure 2.1. Costs in the supply chain, *(Frazelle, 2002)*

Warehousing operations can be listed as receiving, storing, order picking and shipping with corresponding proportional costs illustrated in Figure 2.2.

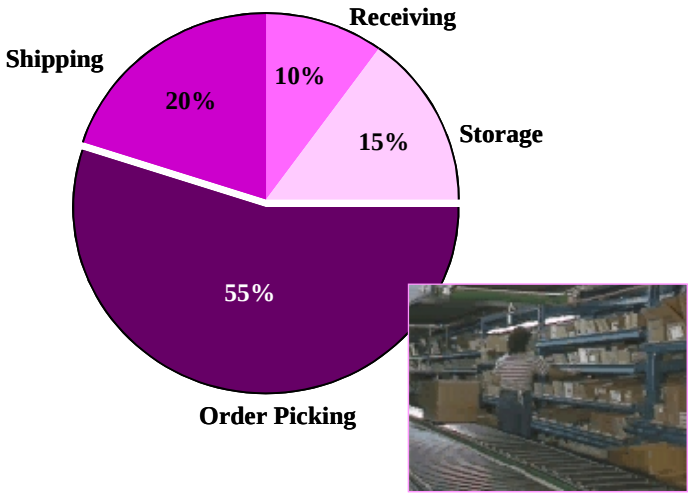


Figure 2.2. Operational costs in a warehouse, *(Frazelle, 2002)*

Order picking constitutes the largest time component of the total operational costs in a warehouse (van den Berg 1999). As depicted in Figure 2.3, among the components of total order picking time, travelling constitutes the biggest proportion in comparison with the other components such as (van den Berg 1999): searching, extracting, documentation, and other activities.

Because order picking and consequently travelling is time consuming operations, reducing the travel time spent on order picking will enable an important reduction in terms of operational costs in a warehouse.

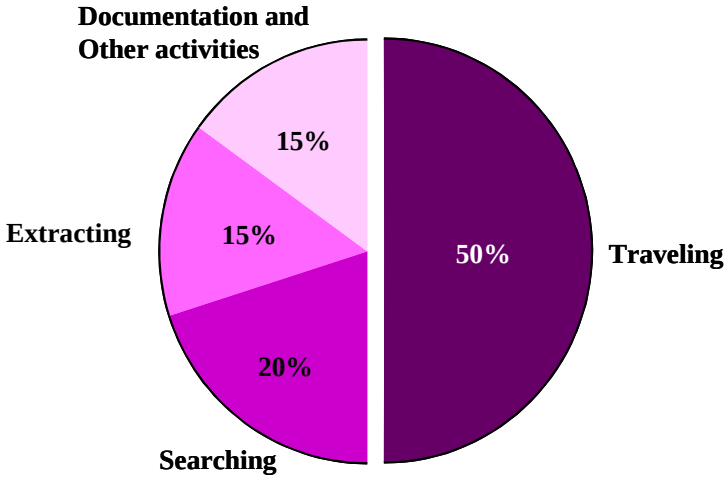


Figure 2.3. Order picking time components, *(Frazelle, 2002)*

2.1.1. Order Picking

In warehouses and distribution centres, products stored in specified locations have to be picked according to customer orders. Order picking is the most time consuming activity among warehousing operations and can be affected by the layout of the facility, by the storage retrieval system, by the zoning, batching, and routing strategies.

Recent research papers on warehousing reflect a wide diversity concerning these strategies: Rouwenhorst et al. (2000) classify types of warehouses design and operating dilemmas in a reference framework. They argue for design-oriented studies, instead of the current separate analysis-oriented research on warehousing issues.

Sharp (2000) summarizes functional warehouse operations, database considerations, and tactical, strategic and operational factors in warehouse operations. The concentration of this research is on efficiency improvement efforts for order picking. Previous work focused on order picking routing strategies and order batching strategies in order to reduce travel time are summarized as follows:

Ratliff and Rosenthal (1983) suggest an algorithm, based on the number of storage aisles in order to solve the order-picking problem most efficiently. They state the computational time required for their algorithm is linearly dependent upon the number of aisles and time efficient for a realistically size warehouse.

Goetschalckx and Ratliff (1998) develop an efficient optimal algorithm that provides policies up to 30% savings in travel time in comparison to other policies. Unless the pick densities are greater than 50%, in most practical aisle widths, traversal policies (to pick both sides of the aisle in the same pass) are more efficient than return policies (pick one side and then pick the other side).

De Koster and van der Poort (1998) establish the problems of order picking routes in two different warehouse environments: conventional and modern. These two warehouses differ in the depository points. The modern warehouse has a decentralized depot, whereas the conventional warehouse has centralized depot. They claim that decentralized depot is common in warehouses where warehouse management systems (WMS) are in application. In order to find shortest order picking routes for both warehouse environments, they modify the algorithm of Ratliff and Rosenthal (1983), which is originally applied to central depot warehouses.

Because the problem is solved mainly with S-Shape heuristic, which ensures that order pickers follow a S-Shape walking pattern to collect items along the pick locations, de Koster and van der Poort create a new heuristic algorithm to discover the alternative to S-shape heuristic by three realistic order picking environments: (1) the central depot, (2) the decentralized depot, (3) the narrow-aisle high-bay pallet warehouses. Travel time is reduced between 7% and 34% with this algorithm. The authors observe that any improvements in order picking travel time are due to warehouse layout and operations.

De Koster et al. (1999) introduce methods to improve the efficiency of manual order picking activities in the distribution center of De Bijenkorf in Netherlands, a retail chain, where products for 7 subsidiaries are consolidated. The authors report that significant reductions on travel time and efficiency on order picking activities are possible to obtain by applying some fundamental strategies: They apply a routing heuristic, which ensures order pickers to pick items from the both sides of storages. It

results in %30 reductions in travel distance and 1.2 people in the number of order pickers. Additionally, they combine order batching, time-savings method and the combined routing heuristics (De Koster and Van der Poort (1998)) to increase the efficiency of the order picking activities. As a result, %68 improvement on order picking travel distance and a saving of 3 to 4 pickers are achieved.

Studying results on order picking efficiency by locating cross aisles in a warehouse, Vaughan and Petersen (1999) argue that cross aisles offer shorter order picking travel distance because of the flexibility they provide by routing order pickers. However, the researchers warn that when the number of cross aisle becomes excessive, their efficiency declines as the cross aisles constitute additional distance to travel to reach the desired storage locations.

The authors develop a shortest path pick sequencing model that allows for any number of cross aisles in the warehouse. The optimal routing is computed for a large number of randomly generated picking requests, over a variety of warehouse layout and order picking parameters. The results demonstrate that when the main storage aisle length (T) is small, an excessive number of cross aisles can increase average picking travel distance, especially when the number of storage aisles (M) is small, and when pick density is very small or very large. Moreover, the optimal number of cross aisles appears to increase as the main storage aisle length (T) increases, as does the relative savings in travel distance at the optimal number. They foresee that cross aisles are most beneficial for very long storage aisles.

Roodbergen and de Koster (2001) analyse the relationship between warehouse layout and average travel time. They consider a warehouse where a single cross aisle divides the parallel storages aisles on the half. The authors investigate whether the middle aisle brings any improvements on order picking efficiency. They offer a dynamic programming algorithm for the shortest order picking routes and claim that if a middle aisle is added to the layout, average order picking time decreases significantly. Arguing that their algorithm is more complex than the algorithm suggested by (Ratliff and Rosenthal, 1983) for rectangular warehouses with two cross aisles, the authors believe that although further extensions to more cross aisles may be possible, they are not practical.

In their other research, Roodbergen and de Koster (2001) foresee several methods to route order pickers in a warehouse where there are more than one cross aisles. They introduce two new heuristics, combined and combined^{+}, as alternative methods to the S-Shape, Largest Gap and aisle-by-aisle Vaughan and Petersen (1999) heuristics, which are already given in the literature.

The authors observe that combined^{+} heuristic is capable of performing best among the five heuristics; the largest gap is efficient in warehouses where there are two non-interior cross aisles and the pick density is low. To compare the performance of the generated heuristics, a branch-and-bound algorithm is constructed. However, they emphasize that optimal algorithms may result in very long computer execution times. Finally, they declare that the performance of the heuristics should be improved or more efficient heuristics should be generated.

Other results in their research can be summarized as follows:

- The S-shape heuristic never has the best performance of the five heuristics
- The largest gap has the best performance in five situations, each of which has a layout with two cross aisles
- The aisle-by-aisle has the best performance in four situations, of which three equal the travel time of the combined and combined^{+} heuristics
- The combined^{+} heuristic gives the best results in 74 of the 80 instances, of which three equal the travel time of the aisle-by-aisle and combined heuristics
- For each individual order, combined heuristic gives a route that is equal to or shorter than the S-shape route because the combined heuristic chooses between traversing entirely the subaisle or returning to the same side of the subaisle, depending on which gives the shortest travel time
- Due to the fact that aisle-by-aisle, combined, and combined^{+} heuristics use the same system of dynamic programming, they are identical for warehouses with two cross aisles (with no interior cross aisles). In situations where there are three or more cross aisles the combined^{+} has the best performance among the heuristics for all situations except one

- For the situations considered in this paper, the size of the gap between the combined$^+$ and the optimal algorithm varies between 1% and 25%
- Generally we can say that the gap between the optimal and combined$^+$ tends to be larger if the situation is more complex, that is when there are more aisles and/or more items.

Gibson and Sharp (1992) use computer simulation to compare two new procedures for batching orders in a retrieval system against a baseline procedure. The factors, which they consider are: the travel metric, warehouse representation, item location assignments, number of items per order, and the total number of orders. Their results indicate that the two new procedures, in combination with skewed (ABC) item location assignments can reduce batch tour lengths by up to 44%.

Ruben and Jacobs (1999) state that, in the related literature batch construction heuristics are constructed and tested according to three strategies for assigning individual items to storage space. They develop a simulation of a single hypothetical warehouse and derive results from the simulation. Specifically, they employ a model that is square in travel distance and assume the walk and pick method of order retrieval with sequential one-way travel. They also perform sensitivity analysis on workforce level and batch size. Their results indicate that the methods used for constructing batches of orders and for assigning storage space to individual items can significantly impact order retrieval efforts in warehouses.

Research involving congestion are summarized as follows:

Sharp et al. (1998), Jarvis and McDowell (1991) remark that congestion is an issue that needs to be considered in routing and allocation problems.

Jarvis and McDowell (1991) explain that collecting the most frequently picked items into a few aisles may increase congestion between order pickers and add that congestion will not be a problem if there is only one order-picker in the system. This congestion will apply both to small facilities, and to large facilities, which are divided into zones with only one order picker in each zone. In such systems each picker picks a

part of an order and partial orders are then consolidated into a complete order. Even in systems where multiple pickers concurrently work within the same region, interference between pickers is not a problem if there is sufficient aisle space for passing. Only in multipicker systems with limited aisle space the potential for congestion should be considered. Determining the amount of interference or congestion in these situations becomes a complex problem involving the number of pickers, the shape of the inventory curve, and the size of the facility, as well as its physical characteristics. It becomes necessary to trade off reduced travel distance against delays due to congestion.

Sharp et al. (1998) claim that busy distribution systems also suffer from congestion in the order accumulation/sorting area and/or at the shipping dock. The degree of order completion that a product offers, in the context of a typical daily set of orders, may be used to minimize this congestion, as well as to reduce overall travel in order picking.

Research on item allocation are as follows:

Daniels et al. (1998) consider a warehouse environment where parts may be stored in multiple locations, simplifying replenishment of inventory and eliminating the need to reserve space for each item. In this environment, order picking requires choosing a subset of the locations that store an item to collect the required quantity. Thus, both the assignment of inventory to an order and the associated sequence in which the selected locations are visited affect the cost of satisfying an order. They formulate a model for simultaneously determining the assignment and sequencing decisions, and compare it to previous models for order picking. They discuss the complexity of the order-picking problem and derive an upper bound on the number of feasible assignments. They give several extensions of TSP heuristics to the new problem setting and test a tabu search algorithm experimentally.

Jarvis and McDowell (1991) aim to provide a basis for locating products in an order-picking warehouse such that average order picking time is minimized. They develop a stochastic model to ensure that optimal, rather than *good*, results are obtained. They show that warehouse assignment algorithms may be used to optimally allocate products to locations. These results are used to explore the effects of average order size

and the shape of the inventory curve on order picking efficiency. They developed the necessary and sufficient conditions for optimally locating product in a class of symmetric warehouses. Moreover, they show if the aisles are not symmetrically located about the dock, to assign the most frequent picked aisles to the nearest location will not necessarily minimize the travel distance.

Chew and Tang (1999) present a travel time model with general item location assignment in a rectangular warehouse. They propose the exact probability mass functions that characterize the tour of an order picker and derive the first and second moments associated with the tour. They apply the model to analysing order batching and storage allocation strategies in an order picking system. The order picking system is modelled as a queuing system with customer batching. The results are compared and validated through simulations. The effects of batching and batch size on the delay time are discussed with consideration to the picking and sorting times for each batch of orders.

The authors also provide the necessary conditions on designing warehouses. They explain that at the present time, initial warehouse planning or feasibility studies on warehouse upgrading are based on rough-cut estimates. These rough-cut estimates have resulted in either underestimation or overestimation of the actual need for storage space. In the latter case where the labour and land costs are very expensive, the recommendations will likely point to the use of high capital intensive material handling systems to meet the future needs, if not the current handling requirements. Such a system involves huge capital investment that may not be required in actual operations, and this would unnecessarily increase the distribution cost. Under such circumstances, one would prefer to design for maximum utilization of storage space and at the same time having enough resources to meet the handling requirements for the current and future needs at the most reasonable cost. They claim it is in these conditions that evaluating the alternatives between using conventional warehousing and sophisticated material handling systems becomes very crucial. Hence, there is a need to develop optimisation techniques to aid in warehousing planning.

They advocate that travel time models provide useful estimates to the throughput or handling requirements when evaluating different types of material handling systems

at the initial warehouse planning and claim that the travel time model can provide the necessary information on the cost of handling systems, especially the cost of material handlers used at a certain level of service. They derive the required number of material handlers for a given level of service by transforming the order picking system as a queuing system. They also explain that the recent trend in warehousing systems has indicated a change from storing large volume of few items to small volume of many items and add this is mainly attributed to the shorter product life cycle and product diversification, which have compelled the management to adopting inventory reduction programs such as just-in-time, cycle time reduction and quick response. These programs would require a more accurate, timely and highly productive order picking system, in particular, warehouses that provide repackaging, break bulking and reconsolidating activities.

3. THE MODIFIED SHORTEST PATH MODEL

3.1. Introduction

Efficient order picking depends on a variety of factors, including system layout, storage systems, information systems, and operating strategies. An order consists of a subset of the items stored in a warehouse. Order picking is the process of retrieving products from specified storage locations. When an order is received, the warehouse dispatches an order picker from the shipping area to pick the items and transport them back to the shipping area. Several methods can be used to reduce travel times attributed to order picking. One approach is adding cross aisles to provide flexibility and reduce the travel distance. Most warehouses employing manual order picking are composed of several parallel pick aisles, where order pickers travel from one pick aisle to the other through the cross aisles that are located at the ends and along of the main aisles. Vaughan and Petersen (1999) develop a shortest path order-picking model to evaluate the impacts of cross aisles that divide the storage spaces equally. For routing the order pickers, they employ an aisle-by-aisle heuristic. In this thesis, their shortest path order-picking model is modified in order to reflect the impacts of variable block lengths (cross aisle spacing). Our ultimate goal is to develop better lengths for storage blocks (better spacing between cross aisles) with respect to travel distance measure.

3.2. The Model

The model used in this thesis assumes a rectangular warehouse with M main storage aisles and N interior cross aisles. Counting the cross aisles at the head and rear of the warehouse there are (N+2) cross aisles at total that divide the whole warehouse so that the better savings for various warehouses under a various pick densities is obtained.

The model warehouse used in this thesis is depicted in Figure 3.1. The warehouse consists of a number of blocks, which are the storage spaces between two cross aisles. The movement of an order picker dispatched from the starting depot contains two types of movement: vertical and horizontal movement. The vertical movement is the walk of an order picker along the main aisles, in other words from north to south or vise versa. The horizontal walk is the movement along the cross aisles occurring from west to east or vise versa.

When an order is received an order picker is dispatched from the start point, which is the intersection of M^{th} main aisle and $(N+1)^{th}$ cross aisle and ends his/her route at the finish point, which is the intersection of first main aisle and $(N+1)^{th}$ cross aisle. The route, which has to be followed by each order picker, is to pick all the items in main aisle M, then all the items in main aisle (M-1), …, and at last all the items to be picked in main aisle 1. This type of routing is named “aisle-by-aisle” routing by Vaughan and Petersen (1999) and is also implemented in Roodbergen & de Koster (2001). This policy provides one-way traffic along the cross aisle and main aisles allowing them to be narrower than could be required by two-way traffic policy. Since the main aisles are narrow enough an order picker is able to access storage locations on both sides of the aisle with negligible lateral movements. Under these circumstances the total horizontal movement required to pick an order is assumed to be constant. Although Vaughan and Petersen (1999) claim that minimizing the total vertical distance travelled is sufficient to find the shortest picking path, in this thesis the horizontal distance travelled by order pickers will be added to the total vertical distance.

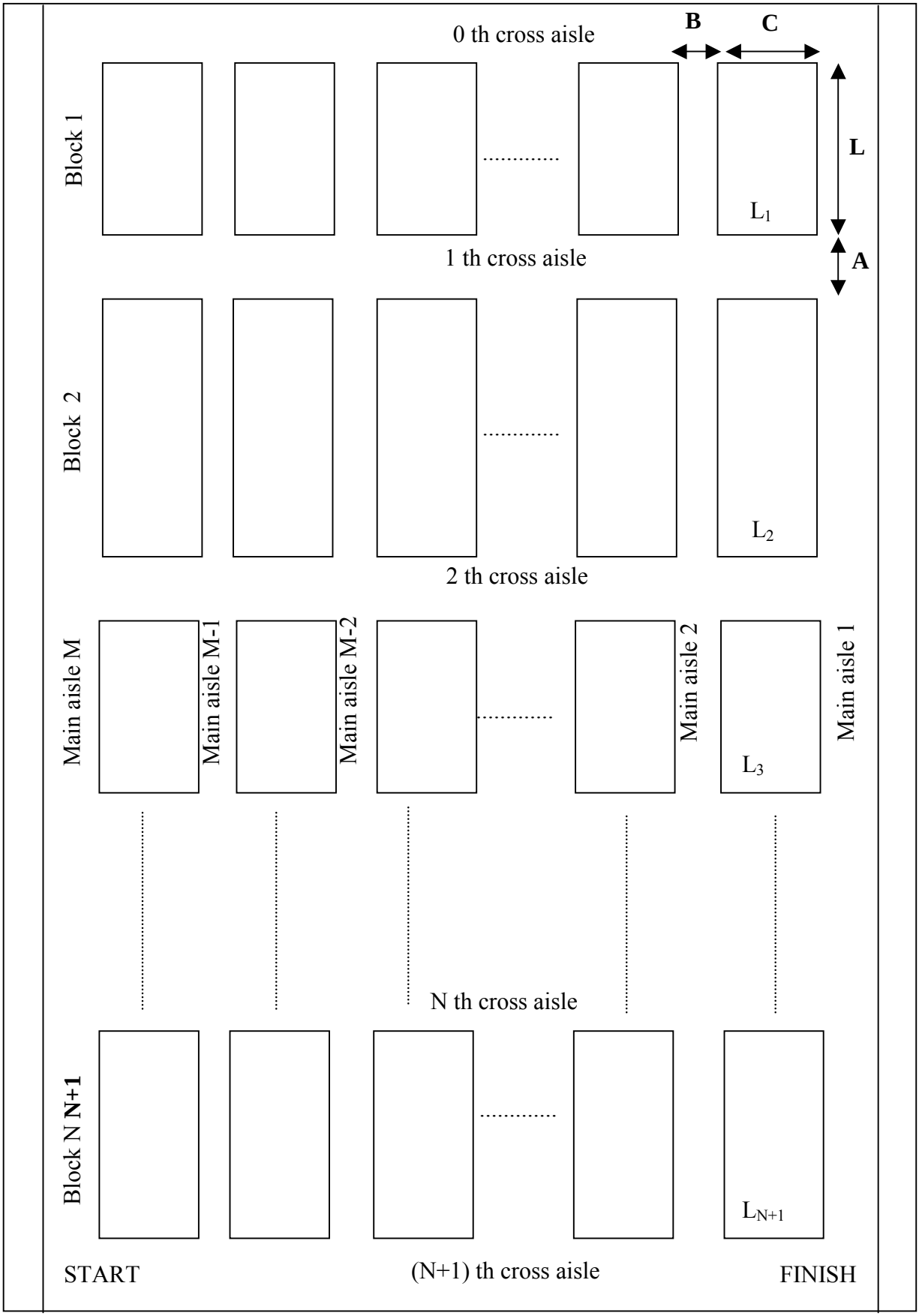


Figure 3.1. The warehouse considered in the research

In this research the number of cross aisles enabling maximum order picking travel distance saving and the best locations for cross aisles are searched for various warehouse settings and order profiles. Total space dedicated to storage is kept constant for a given combination warehouse and order. Since adding a cross aisle to the warehouse layout means a loss of space we assume that the warehouse length is assumed to expand as a cross aisle is added.

In the shortest path order picking model:

- The length of the warehouse (T),
- The number of the main aisles (M),
- The number of the cross aisles (N),
- The width of main aisles (B),
- The width of cross aisles (A)

are assumed to be given and constitute the warehouse settings.

The dynamic programming algorithm developed by Vaughan and Petersen (1999) finds the optimal path to pick an order under the aisle-by-aisle policy are implemented. In this research, search heuristics that try to find the optimal number of cross aisles and their locations. The Java code of this algorithm is presented in Appendix B.

The notation for the shortest path model, which is solved using the dynamic programming algorithm, is as follows (Vaughan and Petersen (1999)):

L_i The length of i^{th} storage block, $i = 1,\ldots, N+1$

T The length of the warehouse equal to the length of main aisles

$$T = \sum_{i=1}^{N+1} L_i$$

M The number of main aisles

N The number of interior cross aisles. (Counting the backward and forward cross aisles located at the head and rear of the warehouse, the total number of cross aisles is (N+2))

A The width of a cross aisle. The width of cross aisles is essential to be addressed in order to evaluate the benefits of cross aisles and their impacts on order picking efficiency. The model considers that horizontal travel distance is constant and an order picker walks along the cross aisles at the centre of the cross aisle. Therefore, any attempt to travel along a cross aisle requires to walk a distance of $A/2$, and any attempt to leave a cross aisle means to walk a distance of $A/2$.

K_m The number of items to be picked by an order picker during a route from the main aisle m, $m = 1, 2, \ldots, M$

$X_m(t)$ The location of an item t in main aisle m, $0 \le X_m(t) \le T$, $m = 1, 2, \ldots, M$, $t = 1, 2, \ldots, K_m$ (undefined if $K_m = 0$)

X_m^+ The location of the item with greatest (south-most) location in the main aisle m (undefined if $K_m = 0$), illustrated in Figure 3.2:

$$X_m^+ = \max_t \{X_m(t)\}$$

X_m^- The location of the item with smallest (north-most) location in the main aisle m (undefined if $K_m = 0$), illustrated in Figure 3.2:

$$X_m^- = \min_t \{X_m(t)\}$$

$Blockof(X_m^-)$ The index of storage block L_i in main aisle m where X_m^- is located, $L_i = 1, 2, \ldots, N+1$

$Blockof(X_m^+)$ The index of storage block L_i in main aisle m where X_m^+ is located, $L_i = 1, 2, \ldots, N+1$

$C_m(i,j)$ The total vertical travel distance required to pick all the items in main aisle m, if main aisle m is entered at cross aisle i, and exited to main aisle m-1 at cross aisle j,

$B1_m(i,j)$ The length of forward-tracking leg required to pick the items in main aisle m to the north of cross aisle h, $h = \min(i,j)$

$B2_m(i,j)$ The length of back-tracking leg required to pick the items in main aisle m to the south of cross aisle h, $h = \max(i,j)$

$f_m(i)$ The minimum total picking distance required to pick all the items in aisle m, m-1, m-2, ..., 3, 2, 1 if main aisle m is entered at cross aisle position i.

The equations for the travel distances are similar to those in Vaughan and Petersen (1999), but include modifications to reflect variable block lengths (cross aisle spacings):

$$C_m(i,j) = B1_m(i,j) + \sum_{s=\min(i,j)+1}^{\max(i,j)} L_s + |i-j|A + B2_m(i,j)$$

where

$$B1_m(i,j) = 2\left[\min\left(\sum_{s=1}^{i} L_s, \sum_{f=1}^{j} L_f\right) - \bar{X}_m + A(0.5 + \min(i,j) - Blockof(\bar{X}_m))\right]$$

$$B2_m(i,j) = 2\left[X_m^{+} - \max\left(\sum_{s=1}^{i} L_s, \sum_{f=1}^{j} L_f\right) + A(0.5 + Blockof(X_m^{+}) - 1 - \max(i,j))\right]$$

3.3. The Dynamic Programming Equations

The dynamic programming equations for each stage are given as follows:

$$f_m(i) = \min_j \{C_m(i, j) + f_{m-1}(j)\}$$
$$f_1(i) = C_1(i, N+1)$$

Stages of the dynamic programming are related to the main aisle numbers in the warehouse. The desired shortest path-picking route is determined by evaluating $f_M(N+1)$. The modified shortest path model is tested and verified with data provided in Vaughan and Petersen (1999).

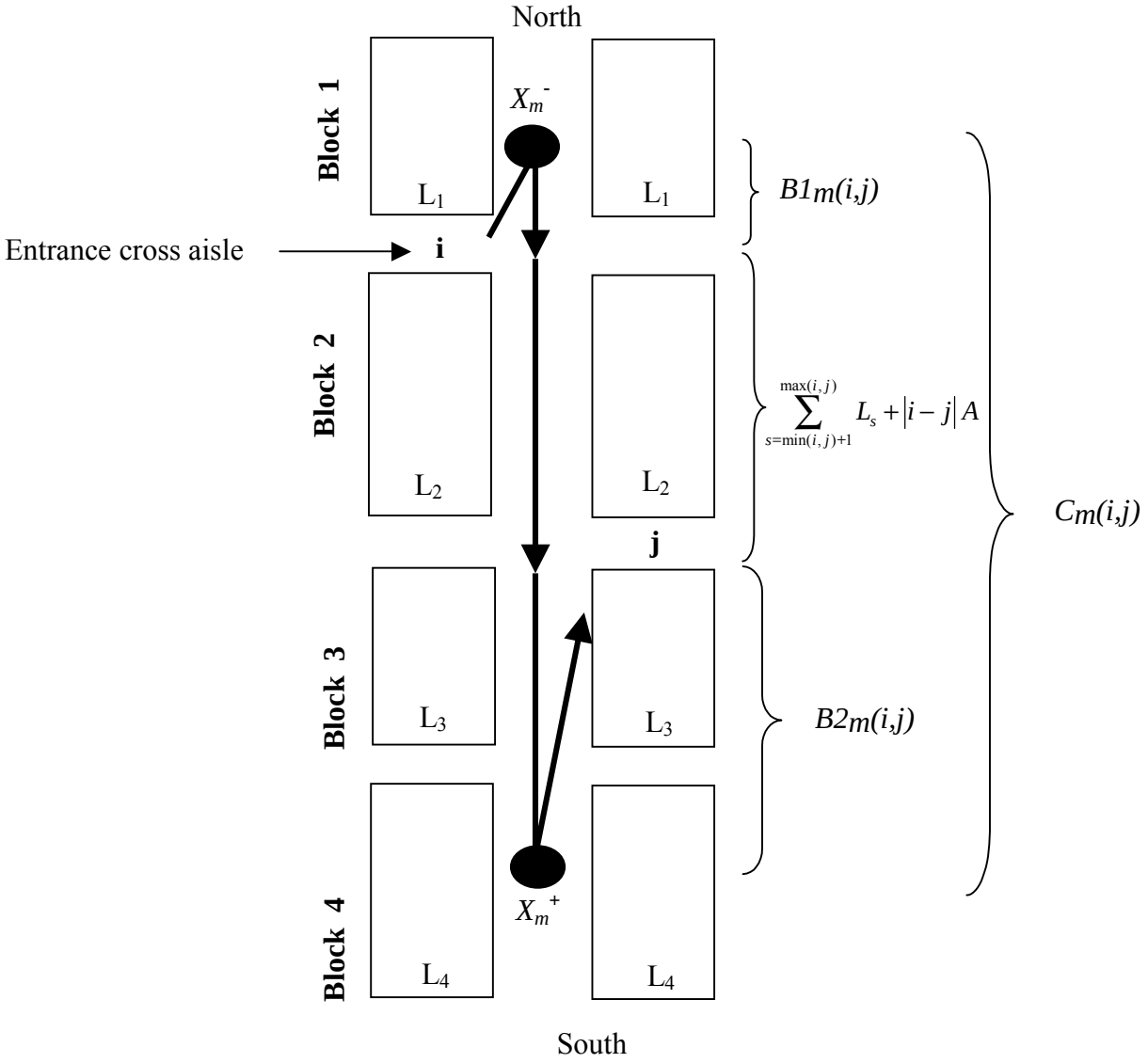


Figure 3.2. The total vertical travel distance to pick items in main aisle *m*, if the main aisle is entered from i^{th} cross aisle and left from j^{th} cross aisle

4. LAYOUT SEARCH ALGORITHMS

Algorithms have been developed in order to generate different feasible configurations of storage block lengths for various types of warehouses. In this thesis, a warehouse is characterized by 6 parameters: warehouse = f (T, M, N, A, B, C). Recall that in Chapter 3, T represents the length of a warehouse, M indicates the number of main aisles available in that warehouse, N is the number of interior cross aisles, A is the width of cross aisles, B is the width of main aisles, and C is the width of storage spaces. Different locations of cross aisles constitute different configurations of storage block lengths. Obviously, when $N = 0$, there is only one storage block.

In this chapter, the layout search algorithms are introduced: the first algorithm to be described, GRID_SEARCH_ALGORITHM, investigates different combinations of grids constituting lengths of storage blocks. The second algorithm, REFINED_GRID_SEARCH_ALGORITHM, improves an initial configuration of storage block lengths.

Number of cross aisles (N) determines how many blocks will be in main aisles. If the warehouse has 2 interior cross aisles, for example, it means that there are 3 storage blocks in that warehouse. The question is how many grids (unit length) should belong to storage block 1, to storage block 2, etc., in other words how long each of them should be in order to obtain maximum travel distance saving.

For this example, the GRID_SEARCH_ALGORITHM would be completed when a set of possible configurations for 3 storage blocks are examined in terms of the saving on order picking travel distance. As indicated above the number of cross aisles N is a parameter of a warehouse. GRID_SEARCH_ALGORITHM investigates a number of possible lengths for storage blocks systematically and designates the configuration of storage blocks that gives the minimum order picking travel distance among tested

feasible storage block lengths. REFINED_GRID_SEARCH_ALGORITHM tries to improve this initial solution for storage block lengths more accurately in order to increase the saving due to the cross aisles.

4.1. Grid and Refined Grid Search Algorithms

In this section, the two search algorithms are presented. In the algorithm pseudo-codes, **bold** words refer to vectors of parameters/variables, *italic* words refer to keywords, "&&" refers to the logical operator "and", and /* */ refers to comments.

4.1.1. Grid Search Algorithm

This algorithm returns **bestL**, the best block lengths among tested layouts, for a given *N*, which is a characteristic of the warehouse.

```
Π = {1,...., N+1}, O = {1,...., θ}

GRID_SEARCH_ALGORITHM (warehouse, orders, noOfGrids)
  G = T/noOfGrids
  for each gridsForL,          /* s.t. gridsForL[i] ≤ noOfGrids, ∀i */
     sumOfGrids = Σ_{i∈Π} gridsForL[i]
     if( sumOfGrids = = noOfGrids&&ARRAY_CONTAINS_NOZERO(gridsForL))
        tempL[i] = gridsForL[i] * G, ∀i ∈ Π
        TempWarehouse.setL(tempL)
        orders[o].setWarehouse(tempWarehouse), ∀o ∈ O
        tempSimulationStatistics = CALCULATE_SIMULATION_STATISTICS(orders)
        tempTravelDistance = tempSimulationStatistics.getAverage()
        if (tempTravelDistance < bestTravelDistance)
          bestL = tempL
          bestTravelDistance = tempTravelDistance
  return bestL
```

CALCULATE_SIMULATION_STATISTICS (**orders**)

travelDistance[o] = getOptimalTravelDistance(**orders[o]**), ∀o ∈ O

return statistics for travelDistance data

The length of the **gridForL** array is (*N*+1) and indicates the number of storage blocks. **gridsForL[i]** records the number of grids that constitute the length of the i^{th} storage block. If the summation of the elements of **gridForL** array is equal to *noOfGrids* value, then a feasible storage block length combination is obtained. When *noOfGrids* = 20 and *N* = 2, for example, then some of the feasible storage block spacing would be as in Figure 4.1.

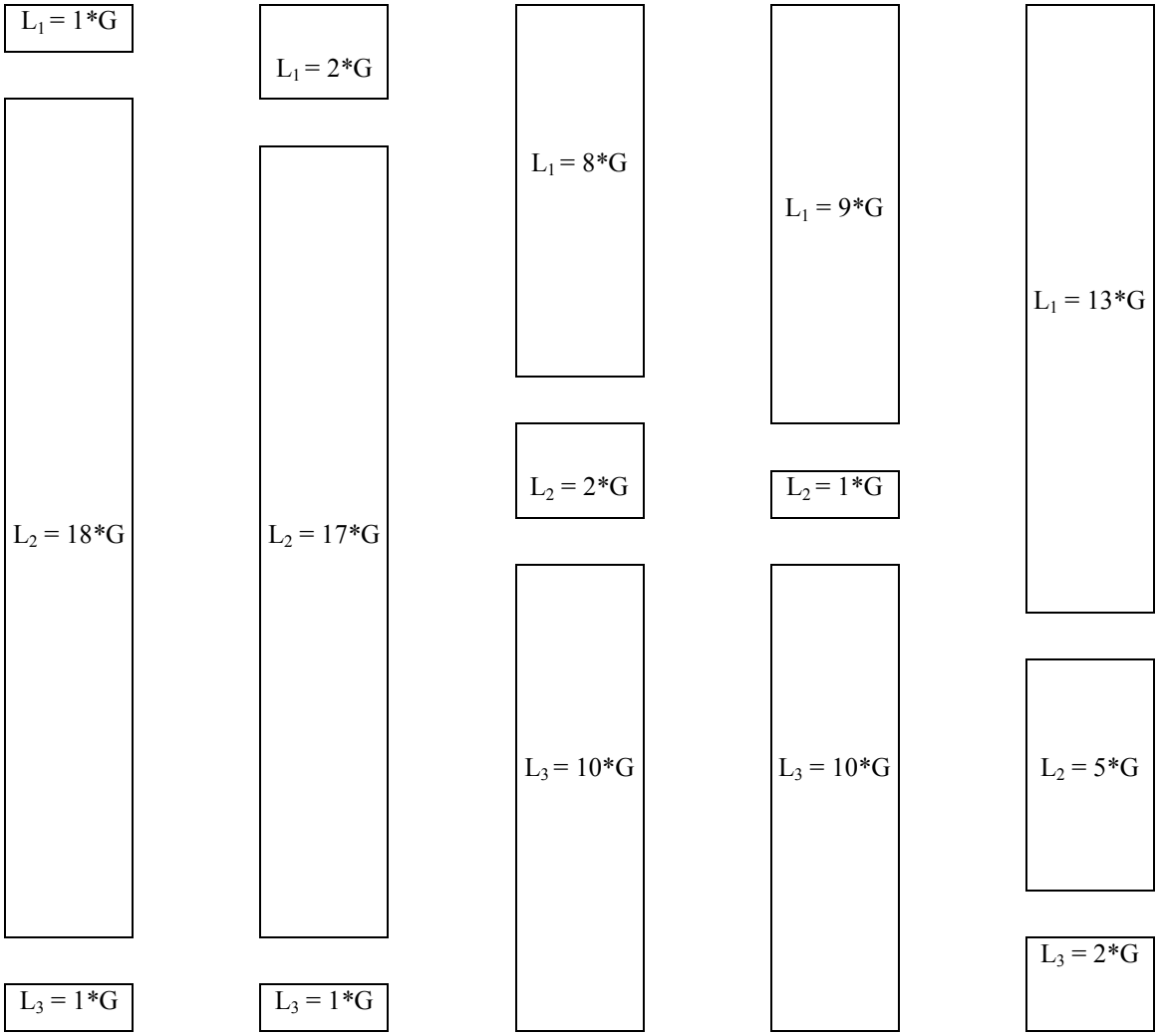


Figure 4.1. Some feasible **gridsForL** configurations

GRID_ SEARCH_ALGORITHM creates all the feasible configuration of storage blocks systematically and returns the traveling distances by solving the modified shortest path dynamic programming algorithm (Vaughan and Petersen (1999)) for the given order set. Average of the travel distances for the order set is taken and the initial best configuration of storage blocks enabling the minimum average order picking travel distances is labeled as **bestL**.

This method generates a great many feasible storage block length alternatives as the number of grid is chosen greater. This results in smaller unit length ($G = T/noOfGrids$). However, the more the number of feasible solution gets, the more will be the computational effort. We observed in our experiments that for the warehouse and order settings described in Chapter 5, *noOfGrids* = 7 was computationally prohibitive (1 week running time including the cases where $N = 5$), and no greater values of *noOfGrids* were used. The Java code of this algorithm is given in Appendix A.

4.1.2. Refined Grid Search Algorithm

```
REFINED_GRID_SEARCH_ALGORITH(warehouse, orders, noOfGrids, resolution)
  initialBestL = GRID_SEARCH_ALGORITHM(warehouse, orders, noOfGrids)
  range = T/noOfGrids
  iterationNo = 0
  continueFlag = true
  while (continueFlag)
      iterationNo++
      if (iterationNo > 1)        // if not the first iteration
          range = (range/noOfGrids)/2
      G = (range/noOfGrids)/2
      for each gridsForL                /* gridsForL[i] ≤ (N+1)*noOfGrids */
        sumOfGrids = Σ_{i∈Π} gridsForL[i]
          if (sumOfGrids==(N+1)noOfGrids&&ARRAY_CONTAINS_NOZERO(gridsForL))
            tempL[i] = initialBestL[i] – (range/2) + gridsForL[i]*G,     ∀i ∈ Π
```

```
        tempWarehouse.setL(tempL)
        orders[o].setWarehouse(tempWaarehouse),    ∀o ∈ O
        tempSimulationStatistics=CALCULATE_SIMULATION_STATISTICS(orders)
        tempTravelDistance = tempSimulationStatistics.getAverage()
        if (tempTravelDistance < bestTravelDistance)
          bestL = tempL
          bestTravelDistance = tempTravelDistance
    if(range<resolution)
      continueFlag = false
return bestL
```

The REFINED_GRID_SEARCH_ALGORITHM starts with the result of the grid search algorithm and applies changes in little unit lengths (*G*) to the initial best configuration of storage blocks (**initialBestL**) to decrease the order picking travel distance for the given order set. In this method, first a *range* is defined. Half of this *range* is subtracted from each storage space length and smaller unit lengths (**gridsForL[i]***G) are added to each storage space length. The travel distance for the new configuration **tempL** is calculated for the given order set (**orders)** and compared with the best result obtained until that time. After trying all feasible configurations of the **gridsForL** for the same initial solution and calculating the travel distance for the new storage block lengths, **tempL** resulting in the shortest travel distance is assigned as best configuration of cross aisles, **bestL**. Then the *range* is updated by dividing with the number of grids (*noOfGrids*). Half of this *range* is subtracted from each storage space length and smaller unit lengths (**gridsForL[i]***G) are added as to obtain new feasible storage spacings (**tempL**) and travel distance implied by the updated **tempL** is calculated for the given order set (**orders**). The refined grid search is continued until the range declines to a length, which is determined as the smallest range (resolution) to be considered. When the range becomes as small as the resolution, the refined grid search is terminated and the improved configuration of storage block lengths is assigned as the best configuration of storage block lengths (**bestL)** for the given warehouse and order set. The Java code of this algorithm is given in Appendix C.

Any element of **gridsForL** can be at most (*N*+1)*noOfGrids, because in the refined grid search algorithm for each storage block, half of the *range* is subtracted and the length **gridsForL[i]*G** is added, for instance:

$$tempL[0] = initialbestL[0] - \frac{range}{2} + gridsForL[0]\frac{range}{2 * noOfGrids}$$

$$tempL[1] = initialbestL[1] - \frac{range}{2} + gridsForL[1]\frac{range}{2 * noOfGrids}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$T = T - (N+1)\frac{range}{2} + sumOfGrids * \frac{range}{2 * noOfGrids}$$

From the above equations it is clearly seen that summation of the **gridsForL**'s elements has to be (*N*+1)**noOfGrids.* Therefore, an element of gridsForL is allowed to be (*N*+1)**noOfGrids* at most.

The search algorithms result in the best storage block lengths (cross aisle spacings) that give the minimum order picking travel distance for a type of warehouse = f (*T*, *M*, *N*, *A*, *B*, *C*) among the tested configurations. Warehouse configurations differ in the values of *T*, *M*, *N*, *A*, *B*, *C*. This procedure is repeated for all of the different warehouse configurations under various order sets to be picked. In the next chapter, Chapter 5, the experimental setting and results are discussed in detail.

5. EXPERIMENTAL DESIGN

The aim of this chapter is to explain the experimental design conducted to establish the best locations of cross aisles for various types of warehouse under various pick densities (order sizes).

For different types of warehouses (warehouse = f (*T*, *M*, *N*, *A*, *B*, *C*)) under various pick densities, best configurations of storage block lengths are investigated. A warehouse is described with 6 parameters which are the length of warehouse *T*, the number of main aisles *M*, the number of cross aisles *N*, the width of cross aisles *A*, the width of main aisles *B*, and the width of storage blocks *C*. Different values of warehouse parameters used in this thesis are depicted in Table 5.1.

Table 5.1. Parameters of a warehouse

Factor	**Number of values**	**Values**
Length of main aisles (*T*)	3	200, 400, 600 (feet)
Number of main aisles (*M*)	3	10, 20, 30
Number of cross aisles (*N*)	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Width of a cross aisle (*A*)	1	10 (feet)
Width of a main aisle (*B*)	1	10 (feet)
Width of a storage block (*C*)	1	10 (feet)

From the cross multiplication of these parameters 90 different warehouse settings are obtained. All these warehouse settings were used for the GRID_SEARCH_ALGORITHM.

However, for the REFINED_GRID_SEARCH_ALGORITHM, N≤5 were used, since it was observed that N≥7 was never to be optimal, and N=6 was to be optimal only once.

Other fundamental parameter used is the pick density (*D*). It is the average number of items to be picked per main aisle and calculated by the ratio of the number of all items to be picked to the number of main aisles in that warehouse. In other words, total number of items to be picked is the multiplication of the pick density (*D*) with the number of main aisles (*M*):

Number of items to be picked = pick density (*D*) x number of main aisles (*M*)

In the experiments, warehouses were tested for different pick density values that generate the number of items to be picked (size of order). 14 different pick densities are applied for each warehouse: 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 5.0 and these densities create orders of varying sizes for each warehouse such that: 0.1*M*, 0.2*M*, 0.3*M*, 0.4 *M*, 0.6 *M*, 0.8 *M*, 1.0 *M*, 1.5 *M*, 2.0 *M*, 2.5 *M*, 3.0 *M*, 3.5 *M*, 4.0 *M*, and 5.0 *M* where *M* is the number of main aisles.

For each warehouse = f (*T*, *M*, *N*, *A*, *B*, *C*) at each order size (0.1*M*, 0.2*M*, 0.3*M*, 0.4 *M*, 0.6 *M*, 0.8 *M*, 1.0 *M*, 1.5 *M*, 2.0 *M*, 2.5 *M*, 3.0 *M*, 4.0 *M*, and 5.0 *M*) three steps are carried out:

1) An order set of 1000 orders is generated by 1000 replication of the following procedure: Each item to be picked is assigned to a random storage location by first generating a main aisle number at random, and then a random number on the interval [0,T) to indicate the position within that main aisle. The locations of the items to be picked in each order are uniformly distributed across the warehouse layout. Assuming item locations to be distributed according to uniform distribution is a common implementation in order picking routing research. (Roodbergen and de Koster (2001))

2) For the given order set and warehouse = f (*T*, *M*, *N*, *A*, *B*, *C*) GRID_ SEARCH_ALGORITHM is applied: For each feasible configuration of storage blocks, the shortest path dynamic programming algorithm (Vaughan and Petersen (1999)) is solved for the order set consisting of 1000 orders. Average of the travel distances for the order set is taken and the initial best configuration of storage blocks enabling the minimum average order picking travel distances is returned.

3) GRID_SEARCH_ALGORITHM returns the initial lengths of blocks which ensures the minimum average order picking travel distance for the given order set at that pick density (*D*). Then, REFINED_GRID_SEARCH_ALGORITHM is applied to the initial best configuration of storage blocks, as described in Chapter 4. This procedure results in the refined best locations of cross aisles (lengths of main aisles), which implies the minimum average order picking travel distance among the tested layouts for the given warehouse under the given pick density (order size).

This experiment is supposed to be repeated for 90 different warehouses at 14 different pick densities. However, some warehouse types are decided to be redundant to investigate according to the results of the initial GRID_SEARCH_ALGORITHM. The above-explained experiment was applied until Step 2 and results were investigated. It was observed that for none except one of the (*T*, *M*, *D*) combinations the best number of cross aisles exceeded 5. This suggests that it is redundant to investigate N>5. This conclusion is supported also with the Vaughan and Petersen (1999). Hence the experiments including REFINED_GRID_SEARCH_ALGORITHM are conducted according to the parameters given in Table 2 for the same pick density levels: 0.1, 0.2, 0.3, 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, and 5.0.

Table 5.2. Parameters of a warehouse in refined grid search algorithm

Factor	**Number of values**	**Values**
Length of main aisles (*T*) (feet)	3	200, 400, 600
Number of main aisles (*M*)	3	10, 20, 30
Number of cross aisles (*N*)	6	0, 1, 2, 3, 4, 5
Width of a cross aisle (*A*) (feet)	1	10
Width of a main aisle (*B*) (feet)	1	10
Width of a storage block (*C*) (feet)	1	10

From the cross multiplication of these parameters 54 different warehouse types are obtained.

The experiments were coded with Microsoft Visual J++ and executed on a HP Workstation x 4000. Execution times were approximately 1 week (168 hours). The code was tested with Rational Quantify and it was observed that the most time consuming part of the code was the dynamic programming algorithm.

6. EXPERIMENTAL RESULTS

The aim of the this chapter is to compare the performance of cross aisles that divide the main aisles in equal lengths and the performance of cross aisles that divide the main aisles in unequal lengths in terms of average order picking travel distance and storage space. The cross aisles in the first type of location will be named as “equally spaced cross aisles” and the second as the “unequally spaced cross aisles”.

The equally spaced cross aisles are introduced by Vaughan and Petersen (1999). Within this thesis the equally spaced case is coded too and validated with the results of Vaughan and Petersen (1999) as explained in Chapter 2. In this thesis the performance of unequally spaced cross aisles will be investigated and they will be compared with the equally spaced cross aisles.

In order to observe the impacts of cross aisles on order picking travel distance, for each warehouse where $N > 0$, the ratio of average order picking distance to the average order picking distance under the $N = 0$ configuration is computed. The ratios less then 1 indicate a saving due to cross aisles, relative to the warehouse layout without cross aisles ($N = 0$).

For each main aisle length (200, 400, 600) and main aisle number (10, 20, 30), different number of cross aisles (0, 1, 2, 3, 4, 5) under 14 pick densities are tested according to the experimental design explained in Chapter 5. It is observed that unequally spaced cross aisles decrease the average order picking travel distance for the given order set more than equally spaced cross aisles do. Moreover, unequally spaced cross aisles enable storage space savings in the warehouse.

We specially focus on the following performance measures:

$$\text{Ratio1} = \frac{\text{Travel distance with cross aisles}}{\text{Travel distance without cross aisles}}$$

$$\text{Ratio2} = \frac{\text{Min. travel distance due to unequally spaced cross aisles}}{\text{Min. travel distance due to equally spaced cross aisles}}$$

$$\text{Ratio3} = \frac{\text{Warehouse length in unequally spaced cross aisles case}}{\text{Warehouse length in equally spaced cross aisles case}}$$

One example to compare the equally spaced cross aisles and unequally spaced cross aisle is the warehouse setting with T = 600 and M = 30. Ratio1 for various pick density (D) values are depicted in Figure 6.1 and Table 6.1. Figure 6.1 gives the maximum travel distance reduction due to equally spaced cross aisles along pick densities, whereas Table 6.1 gives the number of cross aisles providing maximum travel distance reduction for equally spaced cross aisles. When the pick density is 0.1, for example, a Ratio1 of 0.79 is obtained under 6 interior cross aisles.

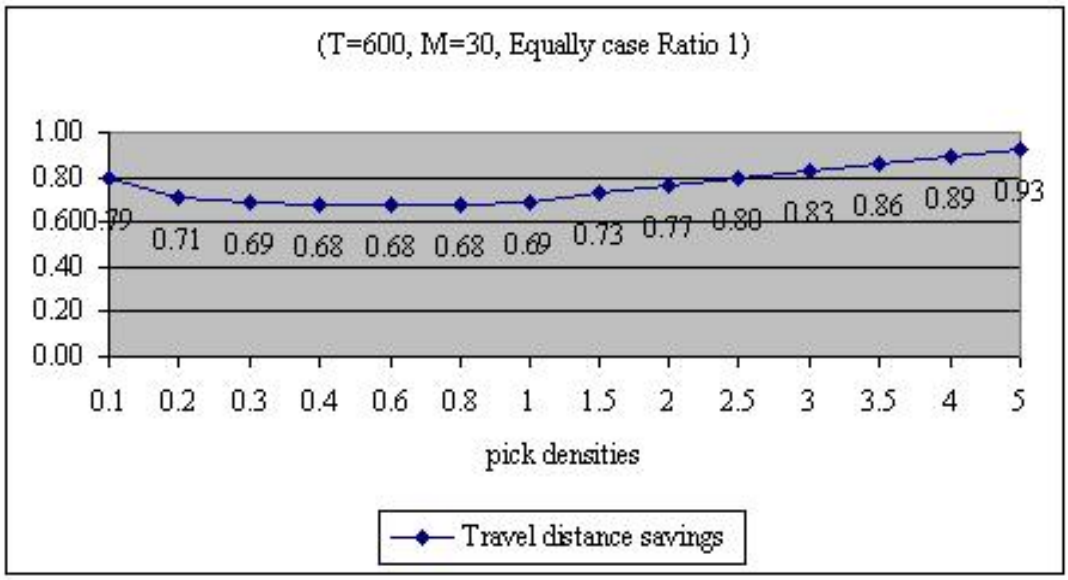


Figure 6.1. Travel distance savings

Table 6.1. Equally spaced cross aisles providing best travel distance reduction

D	0.1	0.2	0.3	0.4	0.6	0.8	1	1.5	2	2.5	3	3.5	4	5
N*	6	6	6	6	6	6	6	6	6	6	5	5	5	5

In Table 6.1, N* refers to the number of cross aisles providing most travel distance savings due to equally spaced cross aisles. As pick density (*D*) increases, N* decreases, which is consistent with Vaughan and Petersen (1999).

Figure 6.2 illustrates the number of unequally spaced cross aisles that provides the same Ratio1 values as in Figure 6.1 provided by equally spaced cross aisles. For the pick density 2.5, for example, Ratio1 = 0.80 is achieved with 6 equally spaced cross aisles, whereas this ratio is gained with 3 unequally spaced cross aisles, whose optimal block lengths are calculated using REFINED_GRID_SEARCH_ALGORITHM.

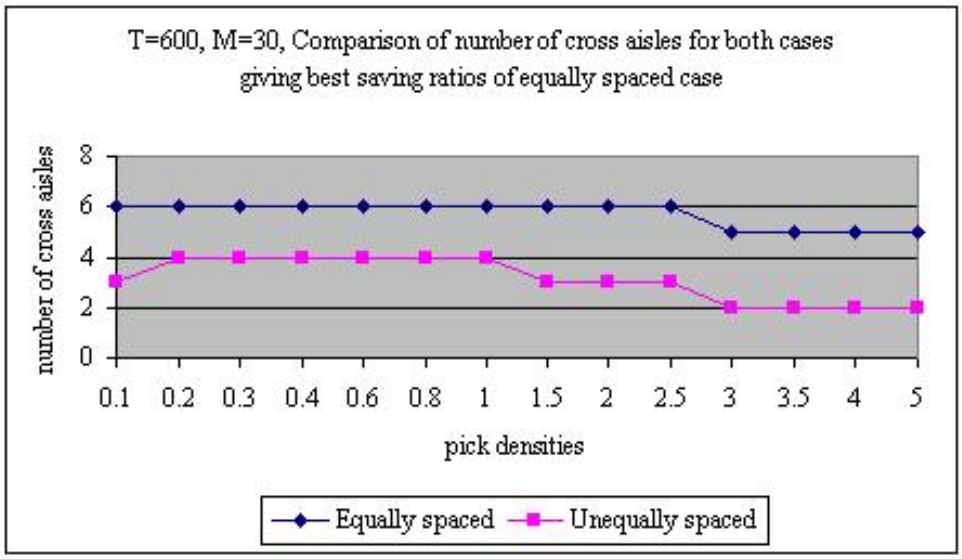


Figure 6.2. Comparison of number of cross aisle

Instead of having 7 equally spaced storage blocks whose length is 85.71 feet, having 4 storage blocks, whose lengths are $L_1 = 91.84$, $L_2 = 153.06$, $L_3 = 238.78$, $L_4 = 116.33$ feet provides the same Ratio1 values. Since adding a cross aisle to the layout of a warehouse constitutes additional cost in terms of space, with unequally spaced cross aisles a reduction on the total storage space requirement is obtained. In rectangular warehouses there are two cross aisles that are on the head and rear of the warehouse and in this thesis the "number of cross aisles" represents the number of interior cross aisles.

So, to find the total number of cross aisles we add 2 more cross aisles to the interior cross aisles. Then the total storage requirement can be calculated with the following formula:

Total storage space = T + (total number of cross aisles) * A

T = length of main aisles, T = 600 feet for the instance warehouse

A = width of a cross aisle, A = 10 feet for all warehouses instances

Adding unequally spaced cross aisles, the maximum travel distance saving due to equally spaced cross aisles is ensured, while the total space requirement is reduced because less number of unequally spaced cross aisles are sufficient. Table 6.2 compares the equally spaced cross aisles and unequally spaced cross aisles that enable Ratio1 values given in Figure 6.1. The storage space saving ratios are given in Table 6.2.

Table 6.2. Storage savings due to unequally spaced cross aisles where T=600, M=30

Pick density (D)	**Equally spaced cross aisles**			**Unequally spaced cross aisles**			**Ratio 3**
	Number of cross aisles	Total number of cross aisles	Total storage space	Number of cross aisles	Total number of cross aisles	Total storage space	
0.1	6	8	680	3	5	650	0.96
0.2	6	8	680	4	6	660	0.97
0.3	6	8	680	4	6	660	0.97
0.4	6	8	680	4	6	660	0.97
0.6	6	8	680	4	6	660	0.97
0.8	6	8	680	4	6	660	0.97
1	6	8	680	4	6	660	0.97
1.5	6	8	680	3	5	650	0.96
2	6	8	680	3	5	650	0.96
2.5	6	8	680	3	5	650	0.96
3	5	7	670	2	4	640	0.96
3.5	5	7	670	2	4	640	0.96
4	5	7	670	2	4	640	0.96
5	5	7	670	2	4	640	0.96

For the presented example, on the average up to 4% storage space reduction (Ratio3 = 0.96) with less number of unequally spaced cross aisles has been obtained. Since the storage space is an important cost component, this result encourages the application of the thesis to real-world situations.

Additionally, with more unequally spaced cross aisles, more travel distance savings can be obtained. In Table 6.3, *N* indicates the number of unequally spaced cross aisles that gives the travel distance saving level due to equally spaced cross aisles. N* represents the number of unequally spaced cross aisles that ensure the maximum saving on travelling distance among all number of cross aisles. For example, in case of the sample warehouse where $T = 600$, $M = 30$, at pick density 5, the level of best reduction due to equally spaced cross aisles is 0.7 and obtained by 2 cross aisles. However, 4 unequally spaced cross aisles enable maximum saving level among all number of equally and unequally cross, 0.11.

Table 6.3. Unequally spaced cross aisles providing travel distance reduction

D	0.1	0.2	0.3	0.4	0.6	0.8	1	1.5	2	2.5	3	3.5	4	5
Unequal N	3	4	4	4	4	4	4	3	3	3	2	2	2	2
Ratio1	0.79	0.71	0.69	0.68	0.68	0.68	0.69	0.73	0.77	0.80	0.83	0.86	0.89	0.93
Unequal N*	3	4	5	5	5	4	4	4	4	3	4	4	4	4
Ratio1	0.79	0.71	0.68	0.67	0.67	0.68	0.69	0.72	0.75	0.80	0.81	0.84	0.86	0.89

Figure 6.3 is the illustration of the Ratio2 values:

$$\text{Ratio2} = \frac{\text{Min. travel distance due to unequally spaced cross aisles}}{\text{Min. travel distance due to equally spaced cross aisles}}$$

Ratio 2 provides a comparison of minimum travelling distances due to unequally spaced cross aisles and equally spaced cross aisles.

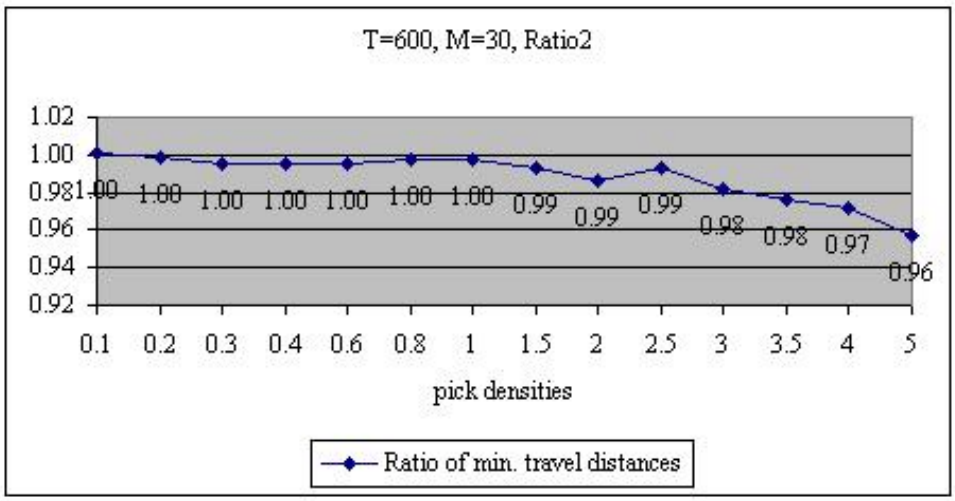


Figure 6.3. Ratio 2 values

As observed in Figure 6.3, unequally spaced cross aisles bring advantage especially at high pick densities in terms of order picking travelling distance.

Although increasing the number of unequally spaced cross aisles provides more order picking travel distance saving as depicted in Table 6.3 and Figure 6.3, the storage space savings weaken due to adding more unequally spaced cross aisles. Results are illustrated in Table. 6.4. When the pick density is 5, for instance, 2 unequally spaced cross aisles give Ratio1 = 0.93 (Table 6.3), whereas it gives Ratio3 = 0.96 (Table 6.2). On the other hand, at the same pick density, 4 unequally spaced cross aisles provide Ratio1 = 0.89 (Table 6.3), but the storage saving ratio it ensures is Ratio3 = 0.99 (Table 6.4).

A decision should be made at this point. If the storage space is more expensive, less number of unequally spaced cross aisles can be preferred, while sacrificing the travel distance saving ratio. However, if the labour costs are more expensive, number of unequally spaced cross aisles can be increased to provide better order picking travel distance saving.

Table 6.4. Storage savings due to unequally spaced cross aisles providing most savings on travel distance

Pick density (D)	Equally spaced case		Unequally spaced case		Ratio 3
	Total # of cross aisles	Total storage space	Total # of cross aisles	Total storage space	
0.1	8	680	5	650	0.96
0.2	8	680	6	660	0.97
0.3	8	680	7	670	0.99
0.4	8	680	7	670	0.99
0.6	8	680	7	670	0.99
0.8	8	680	6	660	0.97
1	8	680	6	660	0.97
1.5	8	680	6	660	0.97
2	8	680	6	660	0.97
2.5	8	680	5	650	0.96
3	7	670	6	660	0.99
3.5	7	670	6	660	0.99
4	7	670	6	660	0.99
5	7	670	6	660	0.99

The plots of the ratios for other settings are given in the Appendix D.

Another interesting issue is the pattern of the storage block lengths. In order to illustrate this pattern following reasoning is made:

For the warehouse where $T = 600$ and $M = 30$, if the number of unequally spaced cross aisles (N*) providing most travel distance saving in Table 6.3 are examined, it is observed that there is a tendency toward four unequally spaced cross aisles. Four unequally spaced cross aisles may be optimal for all pick density levels (D). Table 6.5 gives the comparison of the Ratio1 values due to number of unequally spaced cross aisles providing most saving on travel distance (N*) in Table 6.3 and due to four unequally spaced cross aisles along pick densities (D).

Table 6.5. Comparison of travel distance savings due to cross aisles

Pick density (D)	Number of unequally spaced cross aisles (N*)	Ratio1 values when N* cross aisles are added	Ratio1 values when four (4) cross aisles are added
0.1	3	0.79	0.79
0.2	4	0.71	0.71
0.3	5	0.68	0.69
0.4	5	0.67	0.68
0.6	5	0.67	0.68
0.8	4	0.68	0.68
1	4	0.69	0.69
1.5	4	0.72	0.72
2	4	0.75	0.75
2.5	3	0.79	0.79
3	4	0.81	0.81
3.5	4	0.84	0.84
4	4	0.86	0.86
5	4	0.89	0.89

As depicted in Table 6.5, differences between Ratio1 values due to two alternative number of cross aisles along pick densities (*D*) are not significant, so the number of unequally spaced cross aisles can be selected as 4.

Figure 6.4 gives the lengths of the 5 storage blocks due to 4 unequally cross aisles for the pick density levels. The storage blocks, especially the middle block, reflect a pattern with respect to the pick densities.

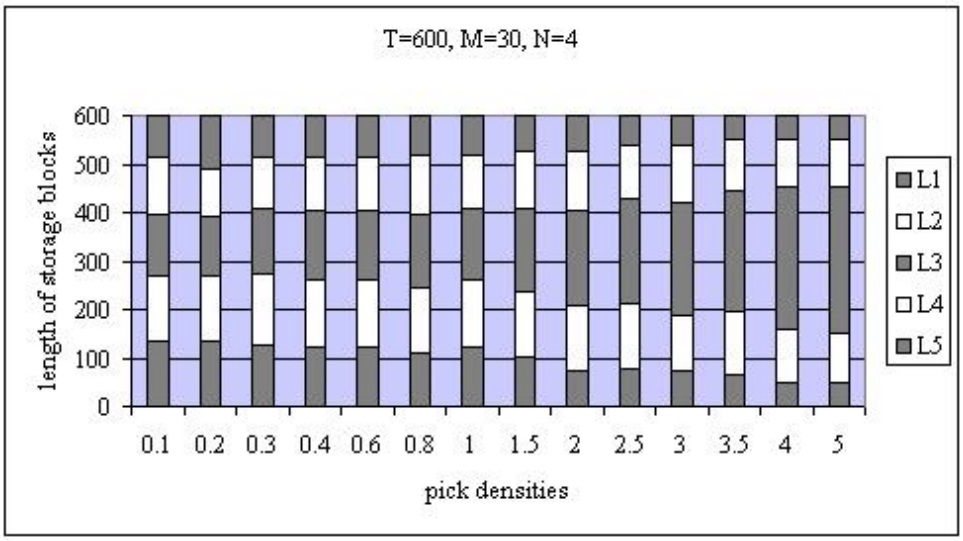


Figure 6.4. Storage block spacing for best number of cross aisles

As it is seen from the Figure 6.4, the middle storage block gets wider as pick density increases. A similar pattern is observed for warehouses with different parameters. The block lengths for some warehouse types are as illustrated in Figures 6.5, 6.6, and 6.7:

In all the following figures *T* is the length of main aisles, *M* is the number of main aisles. *N* indicates the number of unequally spaced cross aisles that provide the maximum travel distance saving in that warehouse.

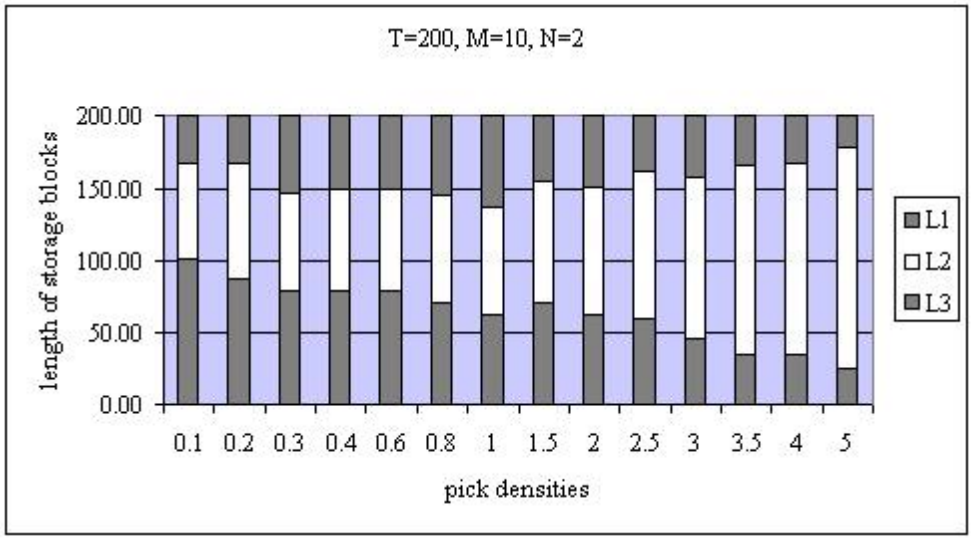


Figure 6.5. Storage block spacing for T=200 – M=10

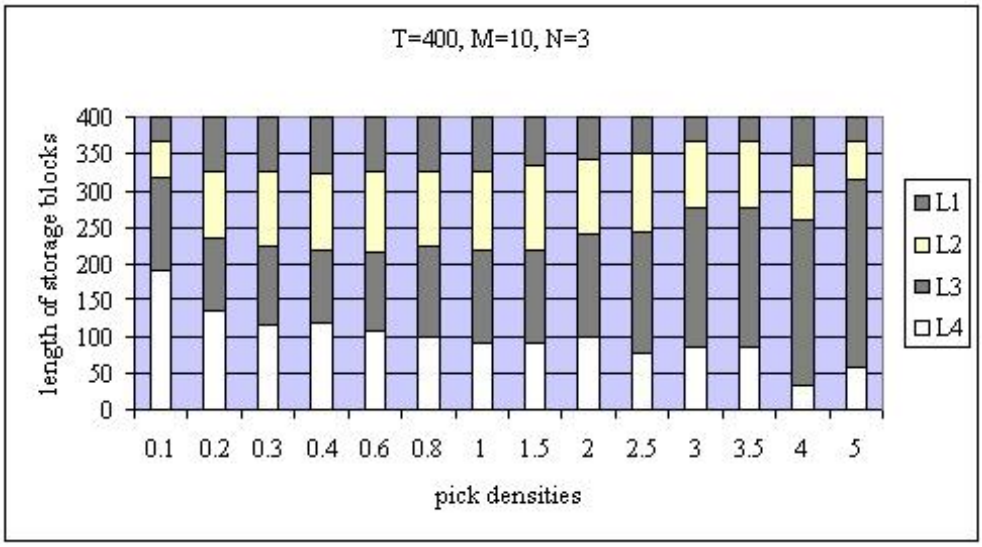


Figure 6.6. Storage block spacing for T=400 – M=10

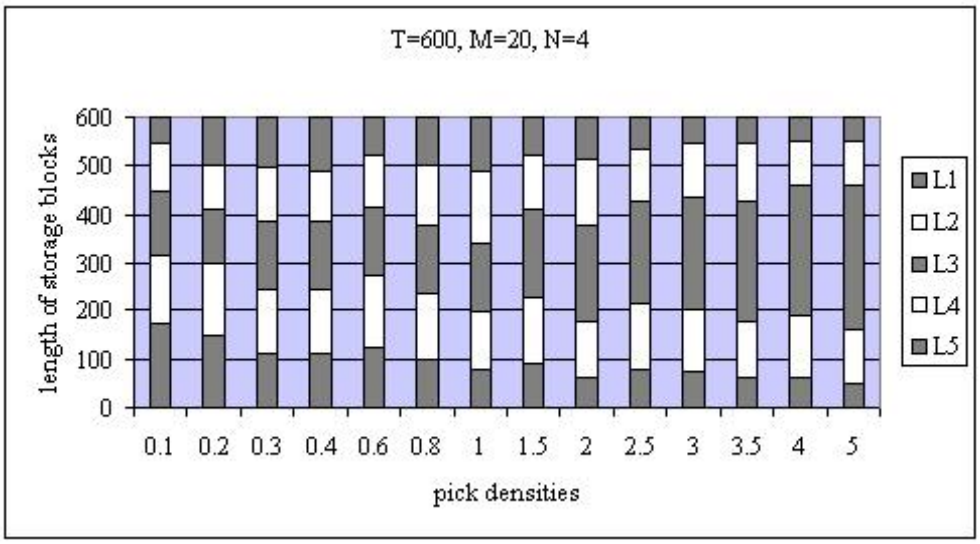


Figure 6.7. Storage block spacing for T=600 – M=20

As the pick density increases the spacing of unequally spaced cross aisles providing maximum travel distance saving appears as depicted in figures. A high pick density means that more items per main aisle is to be collected. Therefore, to have the main aisle in the middle fairly long provides a storage zone, which is undivided. And this strategy avoids the additional distance on the middle storage block that occurs by adding cross aisles and leads to saving on travel distance.

7. CONCLUSIONS AND FUTURE RESEARCH

This research is concerned with order picking efficiency in warehouses. Average order picking travel time in warehouses depend on many factors such that warehouse size, number of main aisles, location of the depot, order size, order picking equipment, order picking policy, and storage assignment rules. This thesis examines the issue of adding cross aisles between storage blocks to improve the order picking efficiency. Research carried out on this issue evaluated the impacts of cross aisles that are located between storage blocks according to even spacing. In this thesis, storage block spacing between cross aisles is focused and the impacts of unequally spaced cross aisles on order picking travel distance are examined.

In this research, two layout search algorithms are introduced. The first, GRID_SEARCH_ALGORITHM examines numerous configurations of storage block lengths (cross aisle spacing) with respect to average travel distance and designated an initial configuration of storage block lengths (cross aisle spacing) resulting in the minimum average travel distance among tested configurations. The latter, REFINED_GRID_SEARCH_ALGORITHM takes the initial configuration and searches for a better configuration of the storage block lengths that decreases the initial minimum travel distance. In order to calculate the average travel distance the dynamic programming algorithm introduced in Vaughan and Petersen (1999) is modified.

Based on the experiments carried out, it is observed that for each warehouse setting (54 value sets) and pick density (14 values) less number of unequally spaced cross aisles provide the same travel distance reductions due to more number of equally spaced cross aisles. Since cross aisles occupy storage space, less number of unequally spaced cross aisles provide saving on warehouse size while ensuring as much travel distance reduction as equally spaced cross aisle ensure. The maximum warehouse space saving due to less number of cross aisles is 4% on average. Moreover, as the number of

unequally spaced cross aisles increase to a certain extent, the travel distance saving that is provided increases too. Their additional travel distance reduction percentage rises up to 4%.

Additionally, an interesting pattern of storage block lengths with respect to pick densities is observed. For each warehouse setting and order size, within the storage block length configuration that provide maximum travel distance reduction, the middle storage block gets wider as pick density increases.

In this research, it is considered that products are distributed uniformly in the warehouse. Other storage assignment rules may be applied, but assuming uniform distribution is a common sense in routing researches. Aisle-by-aisle routing policy (Vaughan and Petersen (1999)) is selected. Although other routing policies could be implemented, in practice the simplicity of the routing policy is vitally important to prevent order-picking errors.

Since warehouse design is a strategic problem concerning long-term investment, the results provided within this thesis provide insights in order to decrease the investment costs or increase the efficiency of warehouse operations.

This research can be extended in many ways:

- The comparison between equally and unequally spaced cross aisles can be made under routing strategies other than aisle-by-aisle. Roodbergen and de Koster (2001) present a total of five routing algorithms with varying complexity. These different algorithms and their interaction with block sizes can be investigated.
- The orders in the thesis are assumed to have random item locations. In practice, popular items assigned to locations closer to the depot, and the orders are picked from mostly locations closer to the depot, as opposed to the entire warehouse.
- The allocation of items to pick locations is a major tactical decision. The decision of strategic determination of block lengths with the tactical determination of item locations can be investigated.
- Congestion impacts of unequally spaced cross aisles can be compared to the congestion impacts of equally spaced cross aisles.

8. APPENDICES

Appendix A. The Java code of GRID_SEARCH_ALGORITHM

The following data is given:

- `T`: length of main aisles
- `M`: number of main aisles
- `N`: number of cross aisles
- `A`: width of cross aisles
- `B`: width of main aisles
- `C`: width of storage blocks
- `D:` pick density
- `_warehouseType:` the type of warehouse
- `noOfGrids:` number of unit lengths
- `range(R):` the length which is going to be divided by grids
- `grid(G):` the unit length that divides the `range`

An initial assignment of the storage block lengths for a specific warehouse under a certain pick density should be accomplished. In this algorithm, the configuration providing minimum order picking distance is labelled as the best storage block configuration.

Assign:

- `N:` number of the cross aisles of the warehouse
- `T`: length of main aisles of the warehouse
- `Range`: length that is going to be divided by storage blocks
- `G`: $\frac{range}{noOfGrids}$, unit length that constitutes storage faces

- *noOfAllPermutations*: no of different combinations of grids
- *counter:* the counter that is compared with *noOfAllPermutations*
- *gridsForL*[*N*+1]: the array of grids that constitutes storage block lengths

1. Find a feasible array of grids, gridsForL:

```
while (counter < noOfAllPermutations)
{
   counter = counter + 1;
   //first of all I will increment the array of gridsForL by 1
   //there are tricks involved
   isIncremented = false;
   digitToIncrease = 0;
   while (isIncremented == false )
   {
           gridsForL[digitToIncrease]= gridsForL[digitToIncrease] +1;
           isIncremented = true;
          if(gridsForL[digitToIncrease]==noOfGrids +1)
           {
                 gridsForL[digitToIncrease]=0;
                digitToIncrease = digitToIncrease +1;
                 isIncremented = false;
            }// End Of if(gridsForL[digitToIncrease]==noOfGrids+1)
   }// End of while (isIncremented == false )
   //now we check whether this particular configuration is feasible
   //we just check if the total noOfgrids occupied will be equal to
noOfGrids
 int sumOfGrids=0;
 for (int i = 0;i<N+1;i++)
 {
    sumOfGrids = sumOfGrids + gridsForL[i];
 }
 // we continue if it is a feasible configuration
```

2. If the configuration of the grids, gridsForL[N+1], is feasible, gridsForL[] is converted to the array tempL[N+1] that represents the lengths of storage spaces and this configuration of storage block is updated in that warehouse and in all orders:

```
if(sumOfGrids==noOfGrids&&intArrayContainsNoZero(gridsForL,gridsForL.length) )
 {
        //now convert gridsForL into tempL vector
        for(int i=0;i<N+1;i++)
        {
               tempL[i]= ((double)gridsForL[i])*G;
        }
        tempWarehouse.setL(tempL);
        tempSumL = calculateSumL(tempL,warehouse);
        tempWarehouse.setSumL(tempSumL);
        for (int o = 0;o < orders.length ;++o)
        {
               orders[o].setWarehouse(tempWarehouse);
        }
```

3. The order picking distance for that configuration of storage blocks is calculated by means of Algorithm 2 and compared with the last minimum traveling distance, `bestTravelDistance`. If the recently calculated distance, `tempTravelDistance`, is less than `bestTravelDistance`, then the result is updated and `tempL` is assigned as the `bestL`, which stores the best configuration of storage blocks:

```
tempSummaryStatistics=calculateSummaryStatisticsForOrders(orders,_warehouseType);
     tempTravelDistance = tempSummaryStatistics.getAverage();
           if (tempTravelDistance < bestTravelDistance)
             {
                          for(int i=0;i<N+1;i++)
                          {
                                  bestL[i] = tempL[i];
                                  bestGridsForL[i]= gridsForL[i];
                          }
                          bestTravelDistance = tempTravelDistance;
             }// End of if (tempTravelDistance < bestTravelDistance)
           }//End                      if                      (sumOfGrids==noOfGrids&&
    intArrayContainsNoZero(gridsForL,gridsForL.length) )
          }// End Of while (counter < noOfAllPermutations)
          return bestL;
     }// End of CalculateOptimalL
```

4. The counter is compared with the number of all permutations:

```
if  (counter >= noOfAllPermutations)
{
  return bestL;
}
else
{
  gridsForL = gridsForL + 1;
}
```

5. Assignment of the initial best locations is done:

```
   initialBestL = bestL;
```

Appendix B. Java Code of the dynamic programming for the shortest path order-picking model

In this algorithm, the cross aisle to be taken in each main aisle that provides minimum distance in the proposed warehouse layout is determined. This algorithm supplies the set of cross aisles to be taken from the starting point to the ending point and the minimum total distance incurred of the suggested route.

The following data is given:

- T : length of main aisles
- M : number of main aisles
- N: number of cross aisles
- A: width of cross aisles
- B: width of main aisles
- D: pick density
- $\_warehouseType$: it defines the type of warehouse
- $X_n[M]$: the array of the minimum locations of the items to be picked in each aisle
- $X_p[M]$: the array of the maximum locations of the items to be picked in each aisle
- $LX_n[M]$: the array of the blocks of the minimum locations of the items to be picked in each aisle
- $LX_p[M]$: the array of the blocks of the maximum locations of the items to be picked in each aisle

1. Take the order set which is generated for that warehouse under a specific pick density:

```
for(orderNo = 0; orderNo < orders.length;orderNo++)
{

        policy=orders[orderNo].returnOptimalPolicy(_optimizationAlgorithmUsed,_wa
        rehouseType);
        _travelDistanceForOrders[orderNo] = policy.getOptimalTravelDistance();
        //For each order distances calculated with the appropriate algotihm are
        stored here
}
```

2. Call the method `returnOptimalPolicy(…)` in order to run the dynamic programming.

3. Retrieve `Xn[]`, `Xp[]`, `LXn[]`, `LXp[]` for that warehouse under that pick density

Assign:

- `N:` determine the number of cross aisles of the warehouse
- `L[]`: determine the length of storage blocks of the warehouse
- K_m`[]`: determine the number of items to be picked from each aisle
- X_n`[]`: determine X_n array for that warehouse under that pick density
- X_p`[]`: determine X_p array for that warehouse under that pick density
- LX_n`[]`: determine LX_n array for that warehouse under that pick density
- LX_p`[]`: determine LX_p array for that warehouse under that pick density
- `A:` determine the width of cross aisles of the warehouse

4. Calculate the distance of entering a main aisle from `i`th cross aisle and leaving from `j`th cross aisle. Start with the `M`th main aisle. If there are any items to be picked in main aisle `M`, take the (`N`+1)th cross aisle as the entrance cross aisle to the `M`th main aisle and calculate the order picking distance for each leaving cross aisle `j` (`j` = 0, 1,.., `N`+1). Then determine the leaving cross aisle j, which satisfy the minimum distance to pick the items in main aisle `M`:

```
for (m=1;m!=M+1;++m)
     {
            if (Km[m -1]>0)
                   {
                   if (m==M)
                   {
                                i=n+1;
                                j=iopt[m-2];
                                minCross=Math.min(i,j);
                                maxCross=Math.max(i,j);
                                sumMinL=0.0;
                                sumMaxL=0.0;
                                for (v=1;v<minCross+1;++v)
                                {
                                       sumMinL=sumMinL+L[v-1];
                                }
                                min=sumMinL;
```

```
                    for (v=1;v<maxCross+1;++v)
                    {
                            sumMaxL=sumMaxL+L[v-1];
                    }
                    max=sumMaxL;


                    if (Xn[m-1]>=min)
                    {
                            B1[m-1]=0.0;
                    }
                    else
                    {
                diff=(minCross - LXn[m-1] );
                B1[m-1]=2.0*((min-Xn[m-1])+A*(0.5+(int)diff ));
                    }
                    if (Xp[m-1]<=max)
                    {
                            B2[m-1]=0;
                    }
                    else
                    {
                         LLXp = LXp[m-1]-1;
                         diff=(LLXp-maxCross);
            B2[m-1]=2.0*(Xp[m-1]-max+A*(0.5+(int)diff));
                    }
                    if (minCross < maxCross)
                    {
                            double OrtaKisim=0.0;
                            for(y=minCross;y<maxCross;++y)
                            {
                                    OrtaKisim=OrtaKisim+L[y];
                            }
                            OrtaKisim=OrtaKisim+Math.abs(i-j)*A;
                            Copt[m-1]=B1[m-1]+OrtaKisim+B2[m-1];
                    }
                    else
                    {
                            Copt[m-1]=B1[m-1] + B2[m-1];
                    }
                    iopt[m-1]=i;
                    jopt[m-1]=j;
}// Km[]>0 and m=M
```

5. Assign the leaving cross aisle `j` for the `M`th main aisle as the entrance cross aisle for (`M`-1)th main aisle and calculate the order picking distance for each leaving cross aisle `j` (`j` = 0, 1,.., `N`+1). Then determine the leaving cross aisle j, which satisfy the minimum distance to pick the items in main aisle `M`-1. Continue this type of calculation until 1th main aisle is reached. If it is the 1th main aisle the leaving cross aisle is assigned as the `N`+1th cross aisle, since the shipping depot is at the intersection of the 1th main aisle and `N`+1th cross aisle:

```
if (m==1)
{       j=n+1;
        jopt[m-1]=j;
}
else
{
        j=iopt[m-2];
        jopt[m-1]=j;
}
for (i=0;i<=n+1;++i)
{
        min=0.0;
        max=0.0;
        sumMaxL=0.0;
        sumMinL=0.0;
        minCross=Math.min(i,j);
        maxCross=Math.max(i,j);
        for ( v=1; v<minCross+1;++v)
        {
                sumMinL = sumMinL+L[v-1];
        }
        min = sumMinL;
        for (v=1; v<maxCross+1; ++v)
        {
                sumMaxL = sumMaxL+L[v-1];
        }
        max = sumMaxL;
        if (Xn[m-1]>=min)
        {
                B1[m-1]=0.0;
        }
        else
        {
        diff=(minCross - LXn[m-1]
};
B1[m-1]=2.0(min-Xn[m-1]+A(0.5+(int)diff));
}
if (Xp[m-1]<=max)
{
 B2[m-1]=0;
}
else
{
   LLXp = LXp[m-1]-1;
   diff=( LLXp - maxCross );
   B2[m-1]=2.0(Xp[m-1]-max+A(0.5+(int)diff));
}
if (minCross<maxCross)
{  double OrtaKisim=0.0;
   for(y=minCross;y<maxCross;++y)
           {
               OrtaKisim=OrtaKisim+L[y];
           }
   OrtaKisim=OrtaKisim+Math.abs(i-j)*A;
   C[i]=B1[m-1]+OrtaKisim+B2[m-1];
}
```

```
                    else
                    {
                    C[i]= B1[m-1]+ B2[m-1];
                    }
                }
                    minC=C[0];
                    mini=0;
                    for (i=1;i<=n+1;++i)
                    {       if (C[i]<minC)
                            {       mini=i;
                                    minC=C[i];
                            }
                    }
                    Copt[m-1]= minC;
                    iopt[m-1]= mini;
            }// end of else for m different from  M
    }// end of if Km[m]>0
```

6. Find the shortest travelling distance for the given warehouse and set of orders. This can be calculated by summing the shortest distances for each main aisle. Record the entrance and leaving cross aisles from each main aisle in order to contribute the route:

```
optimalTravelDistance = optimalTravelDistance + Copt[m-1];
entryCrossAisle = iopt;
exitCrossAisle = jopt;
```

7. Return the shortest travel distance and route as an instance of the VaughanPetersenPolicy class:

```
return new VaughanPetersenPolicy (optimalTravelDistance, entryCrossAisle, exitCrossAisle);
```

Appendix C. The Java Code of the REFINED_GRID_SEARCH_ALGORITHM

In this algorithm, the initial best configuration of storage blocks is examined again with the recursive grid search algorithm. The length of storage blocks are increased and decreased with small unit lengths, grids, which declines in every recursive search. So the configuration of the storage block lengths becomes more precise.

The following data is given:

- *T* : length of main aisles
- *N*: number of cross aisles
- *Grid(G)*: the unit length which constitutes the storage blocks by dividing the *range*
- *range(R):* the length which is going to be divided by unit length (G)
- *Resolution* : the minimum value of the range
- *noOfGrids:* number of grids that constitute storage spaces
- *desiredSumOfGrids:* (N+1) * noOfGrids

1. Calculate the initial best configuration of the storage space using GRID_SEARCH_ALGORITHM:

```
public double[] calculateOptimalLRecursively ( Warehouse warehouse,
                                                Order[] orders,
                                                int noOfGrids,
                                                int _warehouseType)
{
double[]initialBestL=calculateOptimalL(warehouse,orders,noOfGrids,_warehouseType);
```

2. Assign the *range* and the unit length:

```
range = T;
     range = range /(double)noOfGrids;
     while (continueFlag)
     {
            iterationNo++;
            if (iterationNo>1)
            {
     range = (range /(double)noOfGrids)/2;
}
G = ( range/(double)noOfGrids )/2;
```

3. Calculate the desired total number of grids:

```
desiredSumOfGrids = (N+1)* noOfGrids;
```

4. Assign a feasible gridsForL array:

```
long counter = 0;
continueFlag2 = true;
while (continueFlag2)
         {
                //first of all I will increment the array of gridsForL by 1
                //there are tricks involved
                isIncremented = false;
                digitToIncrease = 0;
                while (isIncremented == false && continueFlag2 )
                {
                        gridsForL[digitToIncrease]= gridsForL[digitToIncrease] +1;
                        isIncremented = true;
                        if(gridsForL[digitToIncrease]==noOfGrids*2+1)
                           {
                                gridsForL[digitToIncrease]=0;
                                digitToIncrease = digitToIncrease +1;
                                if (digitToIncrease == N+1)
                                {
                                continueFlag2 = false;
                                }
                                isIncremented = false;
                            }//EndOf if(gridsForL[digitToIncrease]==noOfGrids*2+1)
                }// End of while (isIncremented == false )
                //now we check whether this particular configuration is feasible
                // we check if total noOfgrids occupied will be equal to noOfGrids
                if (continueFlag2)
                {
                        int sumOfGrids=0;
                        for (int i = 0;i<N+1;i++)
                        {
                                sumOfGrids = sumOfGrids + gridsForL[i];
                        }
                        desiredSumOfGrids = (N+1)* noOfGrids;
                        // we continue if it is a feasible configuration
```

5. If the configuration of the grids, `gridsForL[N+1]`, is feasible, subtract the half of the range from the initial lengths of the storage blocks and add the multiplication of the elements of the array `gridsForL[]` with grid length `G` in the order. The new lengths of storage blocks constitutes the `tempL[]` array which is a feasible configuration of storage blocks. The `tempL[]` array is feasible as long as the following equations are satisfied:

```
(range/2)*(N+1) = desiredSumOfGrids*G
desiredSumOfGrids = (N+1)*G
for(int i=0;i<N+1;i++)
{
        tempL[i]=initialBestL[i]-(range/2.0)+((double)gridsForL[i])*G;
}
```

6. This configuration of storage block is updated in that warehouse and in all orders:

```
tempWarehouse.setL(tempL);
tempSumL = calculateSumL(tempL,warehouse);
tempWarehouse.setSumL(tempSumL);
for (int o = 0;o < orders.length ;++o)
{
  orders[o].setWarehouse(tempWarehouse);
}
```

7. The dynamic programming is called for the new configuration, `tempL`, and the order picking distance for the new storage block configuration with the given set of orders under the given pick density is calculated and the average distance for the set of orders is calculated:

```
tempSummaryStatistics=calculateSummaryStatisticsForOrders(orders,_warehouseType);
tempTravelDistance = tempSummaryStatistics.getAverage();
```

8. The order picking distance calculated with the dynamic programming algorithm is compared with the last minimum traveling distance, `bestTravelDistance`. If the recently calculated distance, `tempTravelDistance`, is less than `bestTravelDistance`, then the result is updated and `tempL` is assigned as the `bestL`, which stores the best configuration of storage blocks:

```
if (tempTravelDistance < bestTravelDistance)
  {
        for(int i=0;i<N+1;i++)
        {
                bestL[i] = tempL[i];
                bestGridsForL[i]= gridsForL[i];
        }
        bestTravelDistance = tempTravelDistance;
  }// End of if (tempTravelDistance < bestTravelDistance)
```

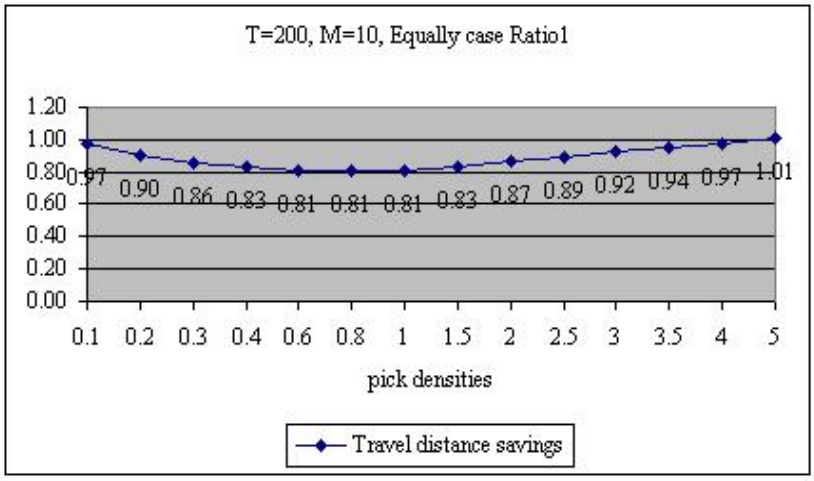
9. In order to calculate other feasible solutions for `gridsForL[]` return to Step 4 and continue. The configuration resulting in the minimum order picking distance is assigned as the final configuration of the storage blocks:

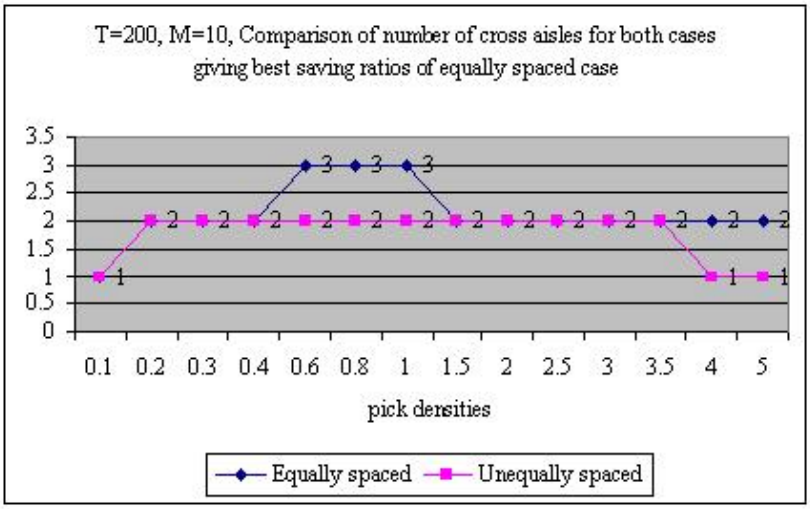
```
for(int i=0;i<N+1;++i)
  {
              finalBestL[i] = bestL[i];
  }
```

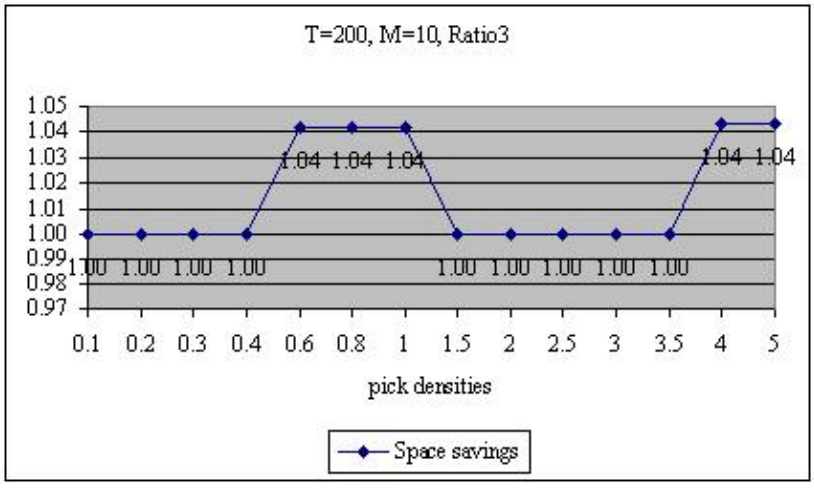
10. `Range` is compared with the resolution, which is the smallest unit to examine. If the range greater than resolution, then the flag is turned to true and range and grid lengths are updated, return to Step 2. If the `range` is less than the `resolution`, then the flag is turned to false and the final configuration of the storage blocks is returned as the result of the recursive grid search:

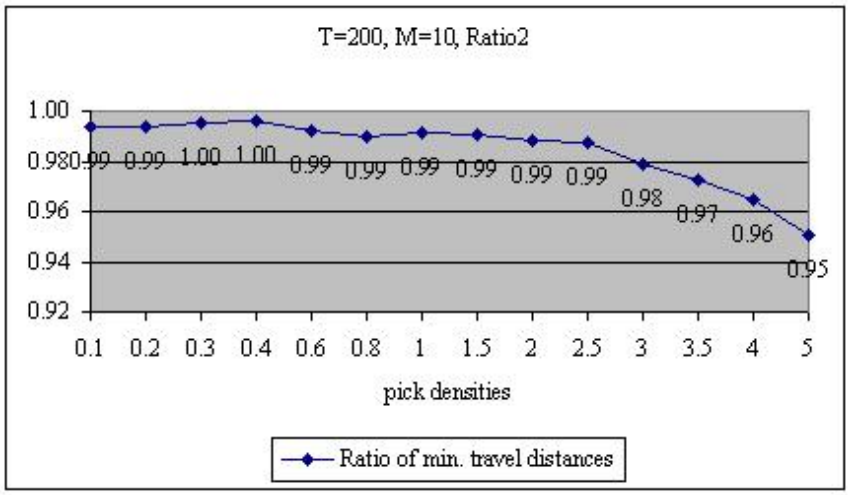
```
        if (range < RESOLUTION)
        {
                continueFlag = false;
        }
  }//while (continueFlag)
        return finalBestL;
}// End of calculateOptimalLRecursively
```

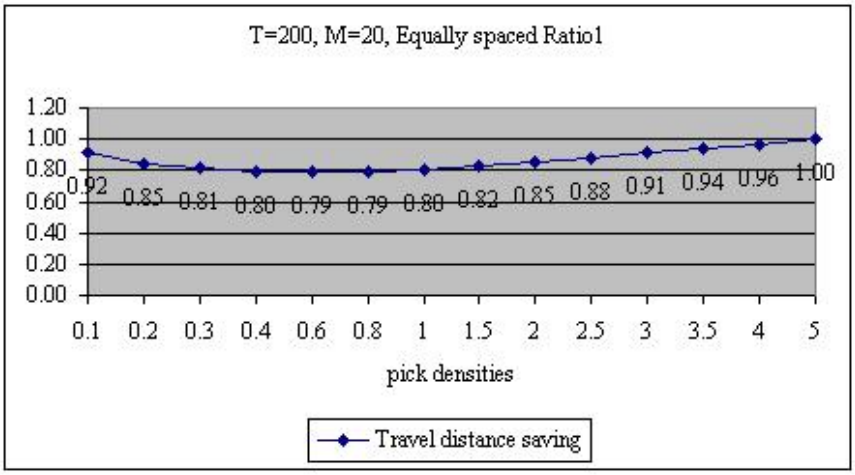
Appendix D.

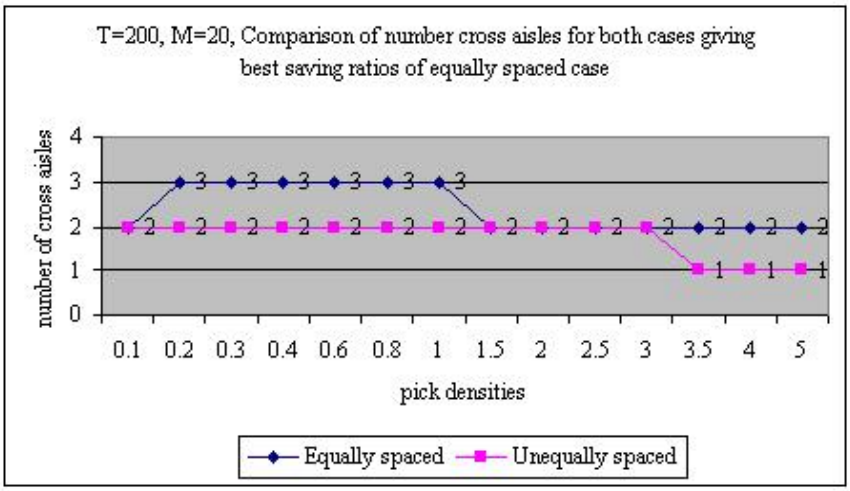

T=200, M=10, Equally case Ratio1
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.97 0.90 0.86 0.83 0.81 0.81 0.81 0.83 0.87 0.89 0.92 0.94 0.97 1.01
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance savings

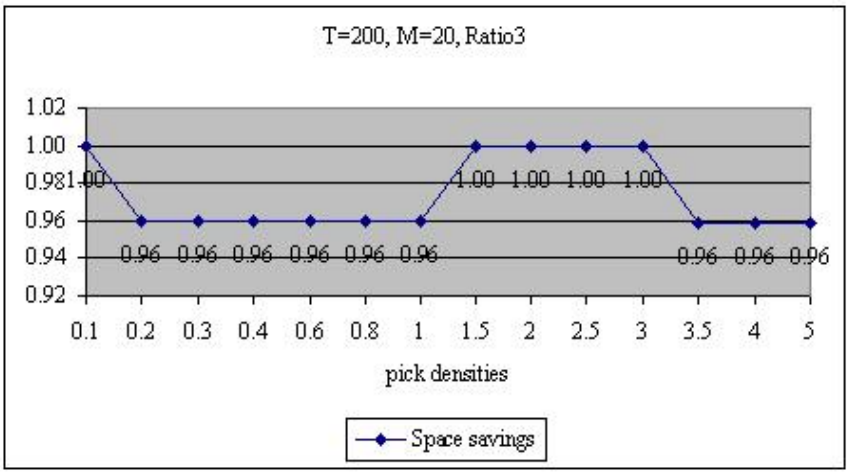

T=200, M=10, Comparison of number of cross aisles for both cases
giving best saving ratios of equally spaced case
3.5
3
2.5
2
1.5
1
0.5
0
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced

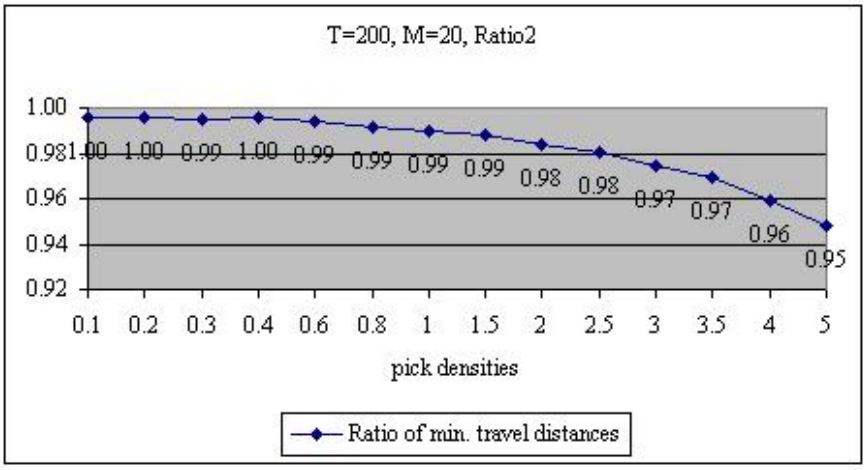

T=200, M=10, Ratio3
1.05
1.04
1.03
1.02
1.01
1.00
0.99
0.98
0.97
1.00 1.00 1.00 1.00 1.04 1.04 1.04 1.00 1.00 1.00 1.00 1.00 1.04 1.04
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings

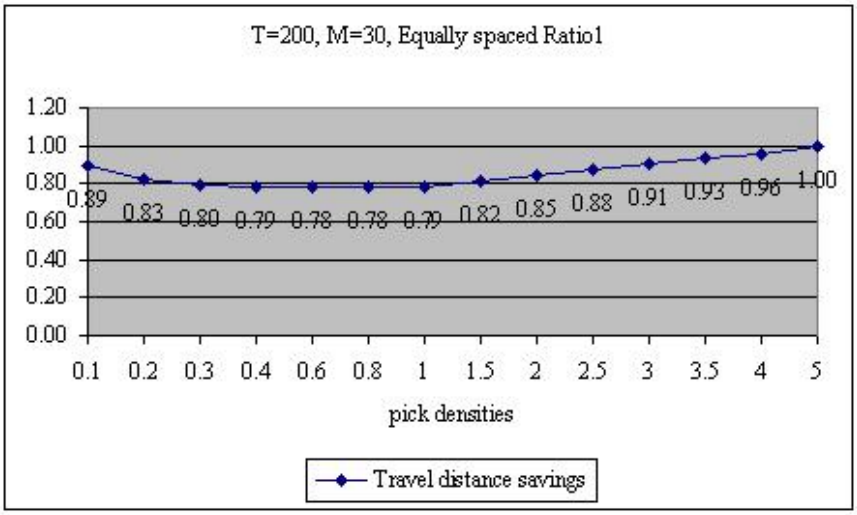

T=200, M=10, Ratio2
1.00
0.98
0.96
0.94
0.92
0.99 0.99 1.00 1.00 0.99 0.99 0.99 0.99 0.99 0.99 0.98 0.97 0.96 0.95
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

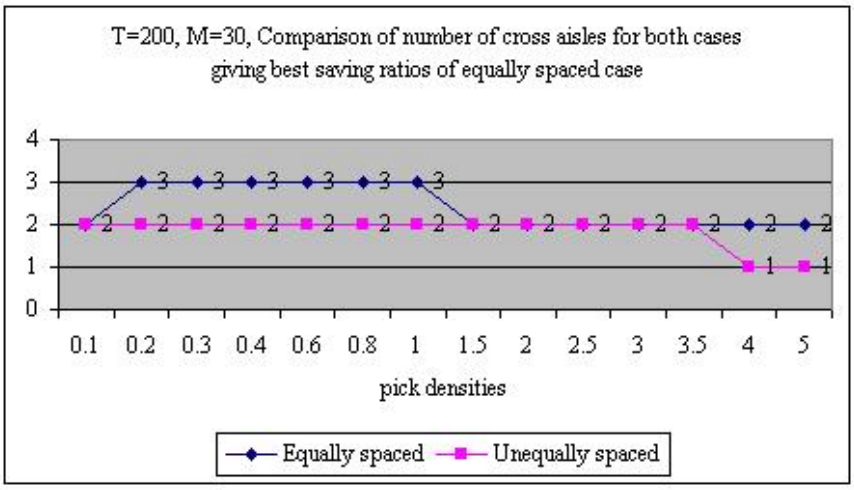

T=200, M=20, Equally spaced Ratio1
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.92 0.85 0.81 0.80 0.79 0.79 0.80 0.82 0.85 0.88 0.91 0.94 0.96 1.00
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance saving

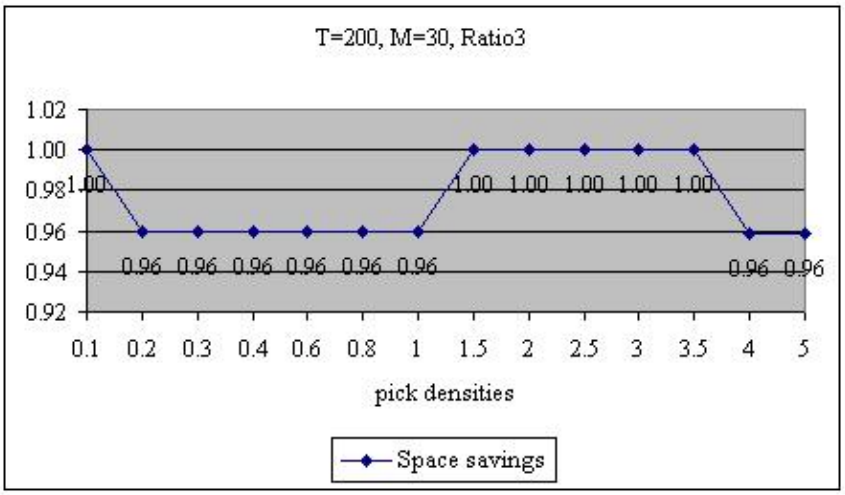

T=200, M=20, Comparison of number cross aisles for both cases giving best saving ratios of equally spaced case
number of cross aisles
4
3
2
1
0
3 3 3 3 3 3 3
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
1 1 1 1
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced

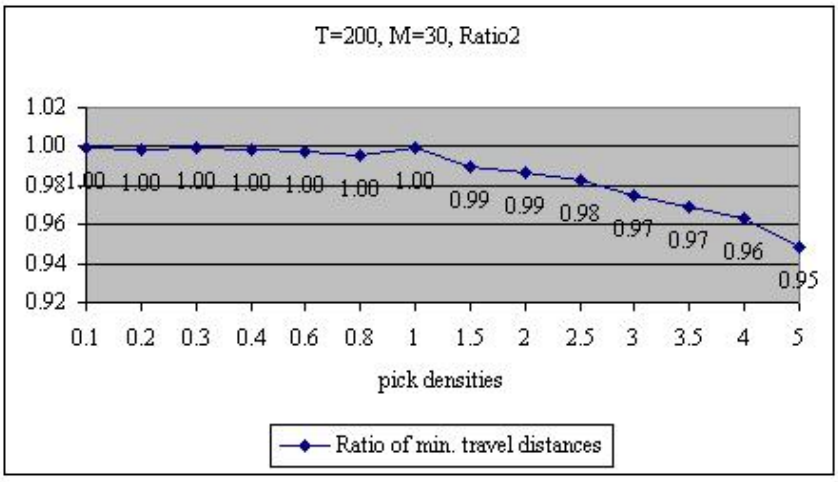

T=200, M=20, Ratio3
1.02
1.00
0.98
0.96
0.94
0.92
1.00 0.96 0.96 0.96 0.96 0.96 0.96 1.00 1.00 1.00 1.00 0.96 0.96 0.96
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings

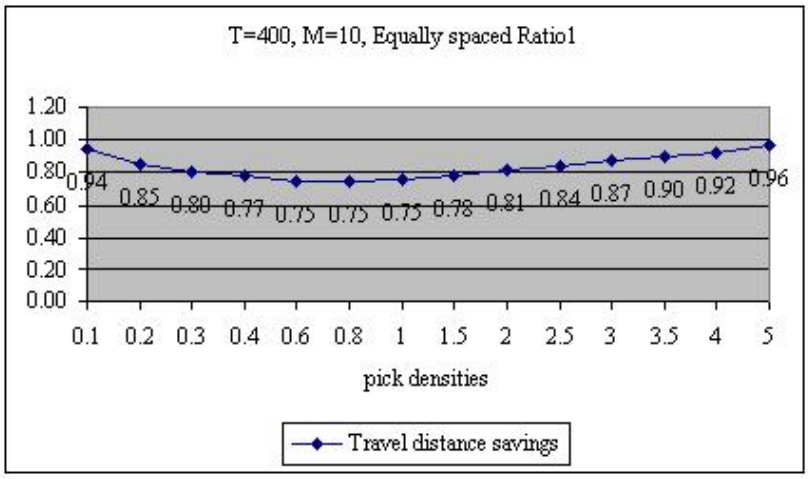

T=200, M=20, Ratio2
1.00
0.98
0.96
0.94
0.92
1.00 1.00 0.99 1.00 0.99 0.99 0.99 0.99 0.98 0.98 0.97 0.97 0.96 0.95
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

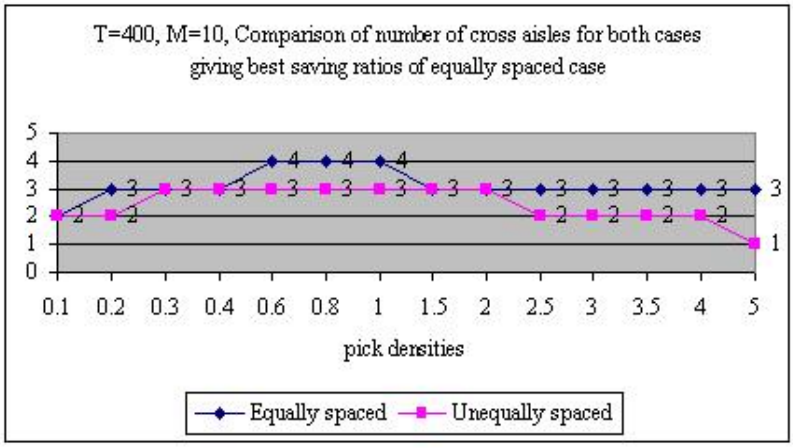

T=200, M=30, Equally spaced Ratio1
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.89 0.83 0.80 0.79 0.78 0.78 0.79 0.82 0.85 0.88 0.91 0.93 0.96 1.00
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance savings

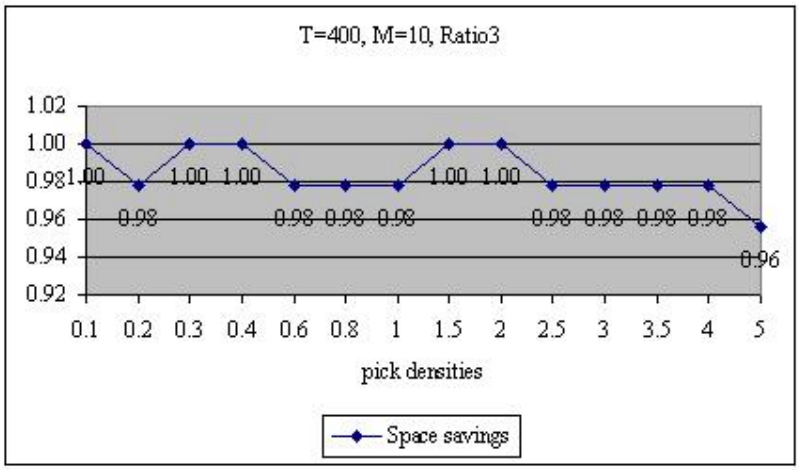

T=200, M=30, Comparison of number of cross aisles for both cases giving best saving ratios of equally spaced case
4
3
2
1
0
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced

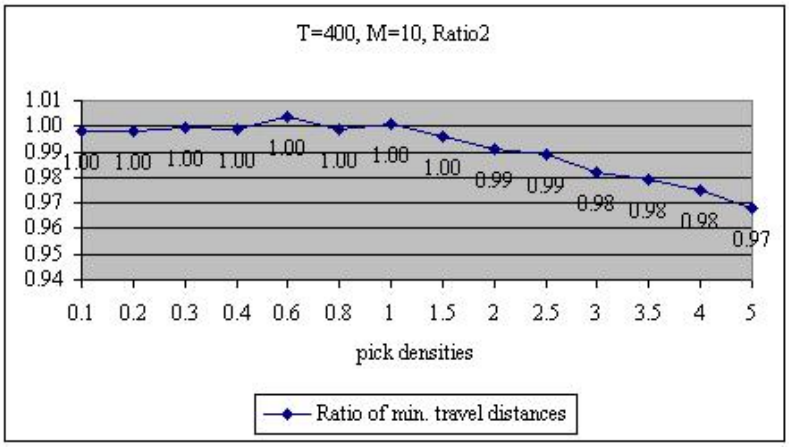

T=200, M=30, Ratio3
1.02
1.00
0.98
0.96
0.94
0.92
1.00 0.96 0.96 0.96 0.96 0.96 0.96 1.00 1.00 1.00 1.00 1.00 0.96 0.96
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings

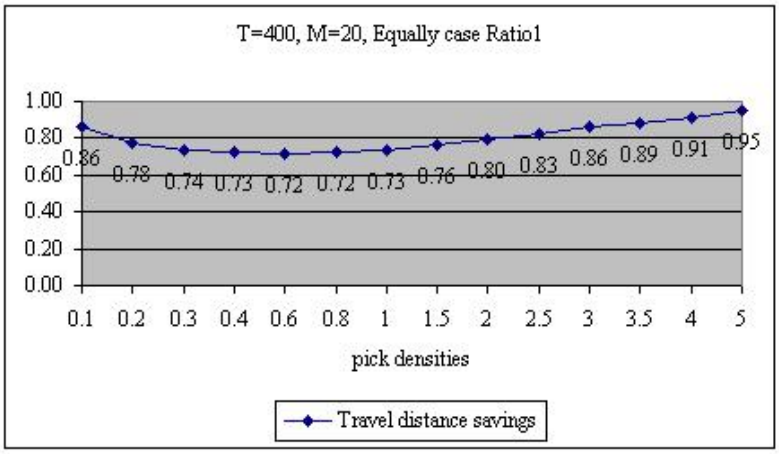

T=200, M=30, Ratio2
1.02
1.00
0.98
0.96
0.94
0.92
1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.99 0.99 0.98 0.97 0.97 0.96 0.95
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

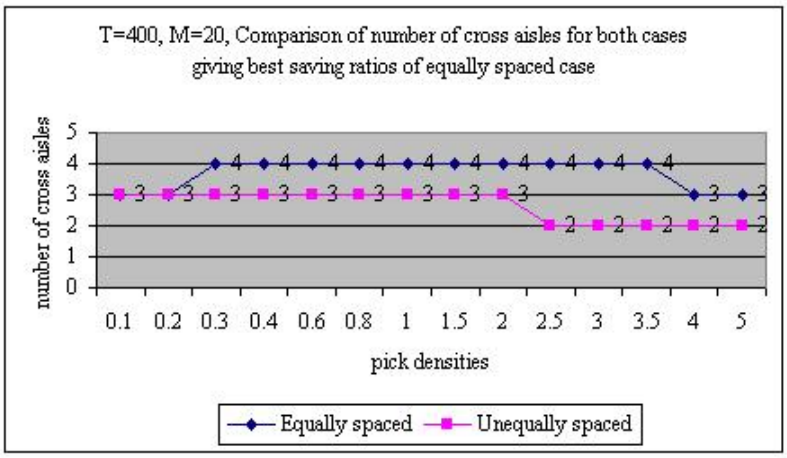

T=400, M=10, Equally spaced Ratio1
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.94 0.85 0.80 0.77 0.75 0.75 0.75 0.78 0.81 0.84 0.87 0.90 0.92 0.96
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance savings

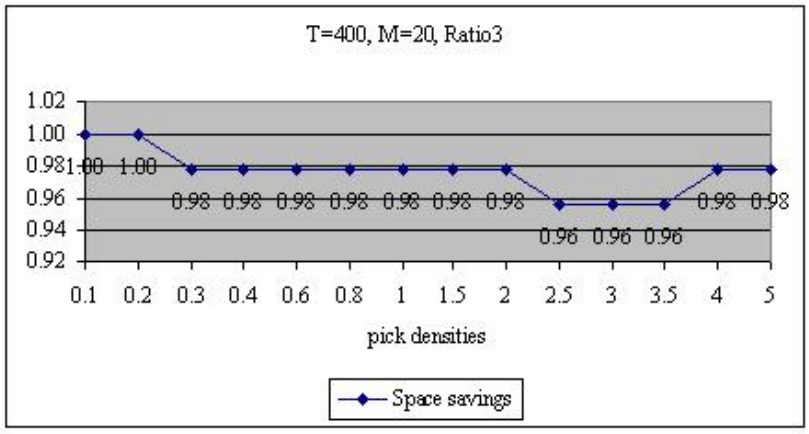

T=400, M=10, Comparison of number of cross aisles for both cases
giving best saving ratios of equally spaced case
5
4
3
2
1
0
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced

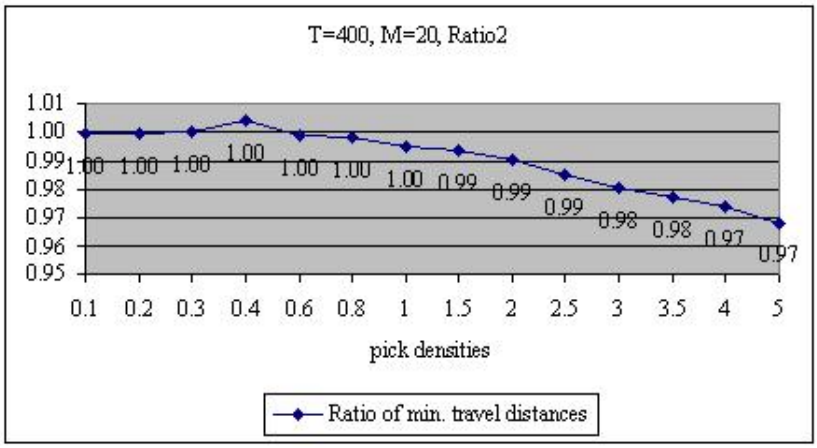

T=400, M=10, Ratio3
1.02
1.00
0.98
0.96
0.94
0.92
1.00 0.98 1.00 1.00 0.98 0.98 0.98 1.00 1.00 0.98 0.98 0.98 0.98 0.96
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings

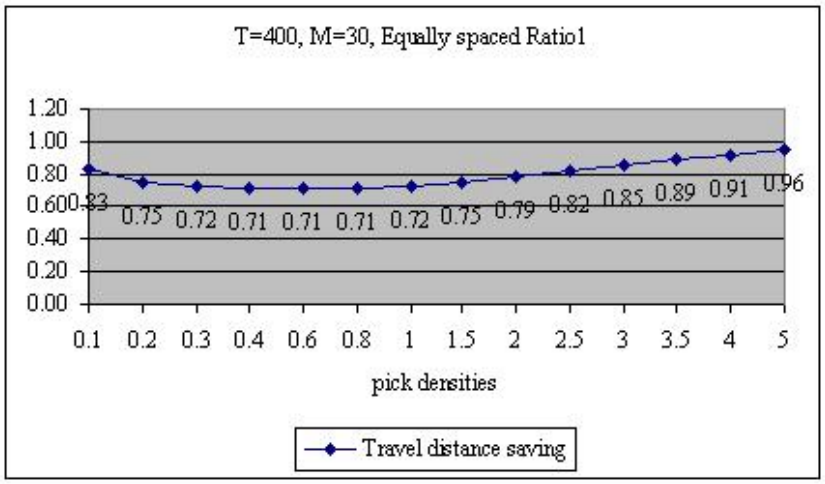

T=400, M=10, Ratio2
1.01
1.00
0.99
0.98
0.97
0.96
0.95
0.94
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.99 0.99 0.98 0.98 0.98 0.97
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

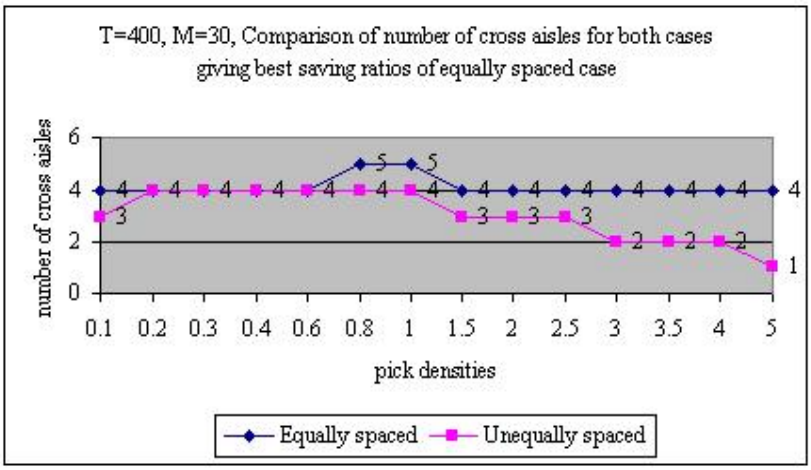

T=400, M=20, Equally case Ratio1
1.00
0.80
0.60
0.40
0.20
0.00
0.86 0.78 0.74 0.73 0.72 0.72 0.73 0.76 0.80 0.83 0.86 0.89 0.91 0.95
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance savings

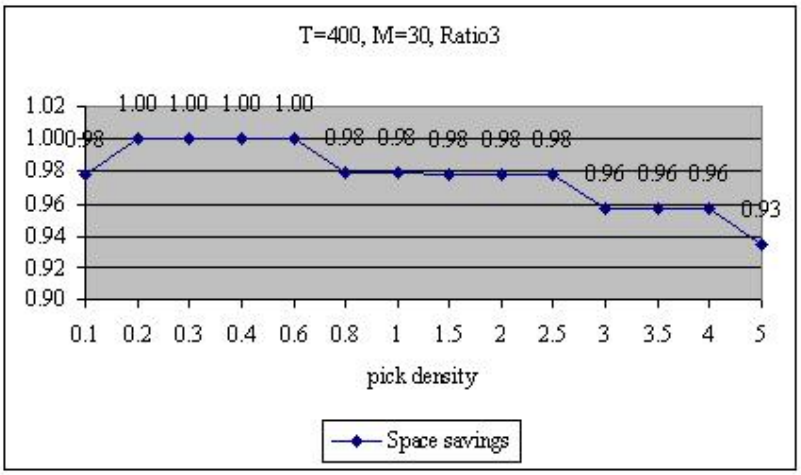

T=400, M=20, Comparison of number of cross aisles for both cases
giving best saving ratios of equally spaced case
number of cross aisles
5
4
3
2
1
0
4 4 4 4 4 4 4 4 4 4 4 4 4 3 3
3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced

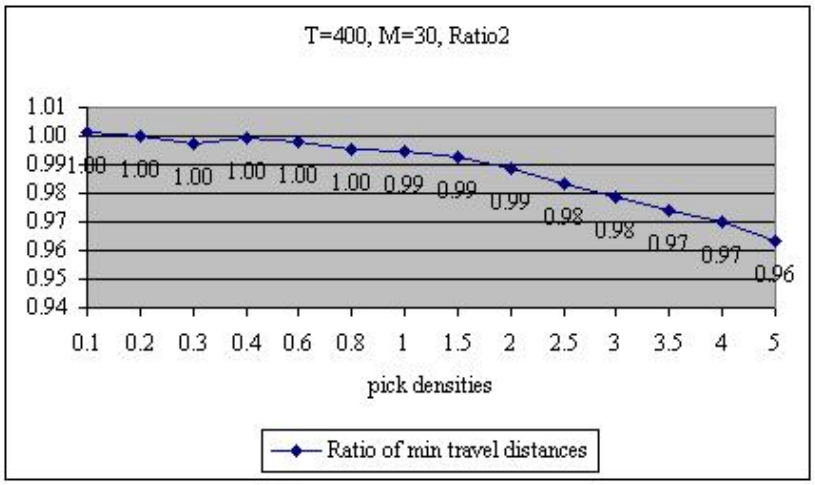

T=400, M=20, Ratio3
1.02
1.00
0.98
0.96
0.94
0.92
1.00 1.00 0.98 0.98 0.98 0.98 0.98 0.98 0.98 0.96 0.96 0.96 0.98 0.98
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings

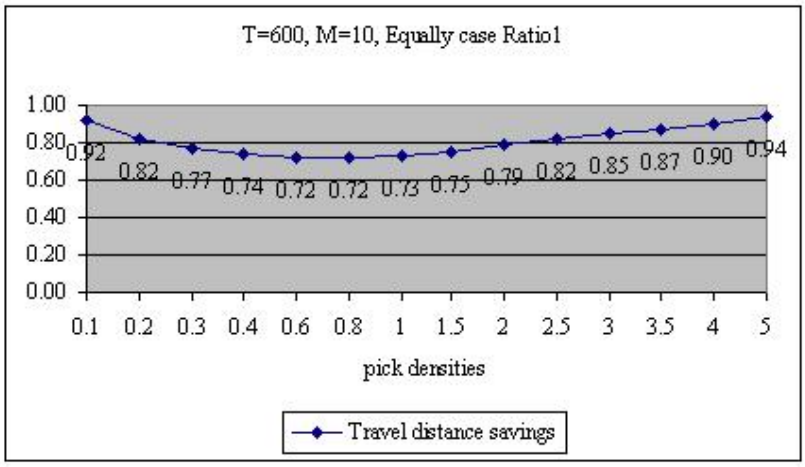

T=400, M=20, Ratio2
1.01
1.00
0.99
0.98
0.97
0.96
0.95
1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.99 0.99 0.99 0.98 0.98 0.97 0.97
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

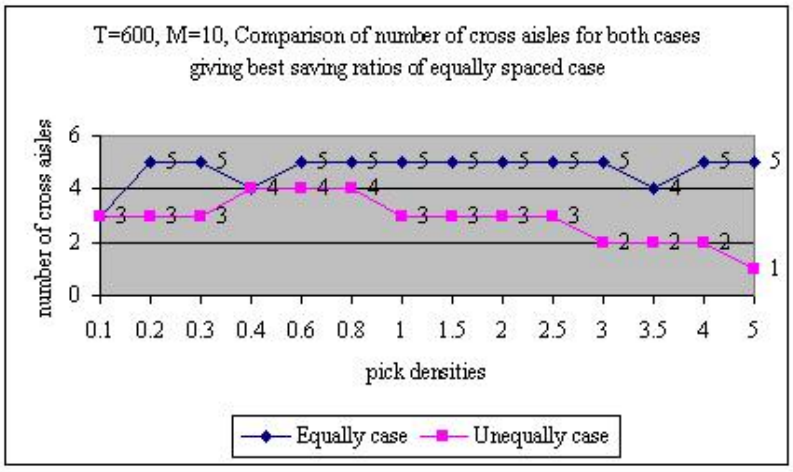

T=400, M=30, Equally spaced Ratio1
1.20
1.00
0.80
0.60
0.40
0.20
0.00
0.83 0.75 0.72 0.71 0.71 0.71 0.72 0.75 0.79 0.82 0.85 0.89 0.91 0.96
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance saving

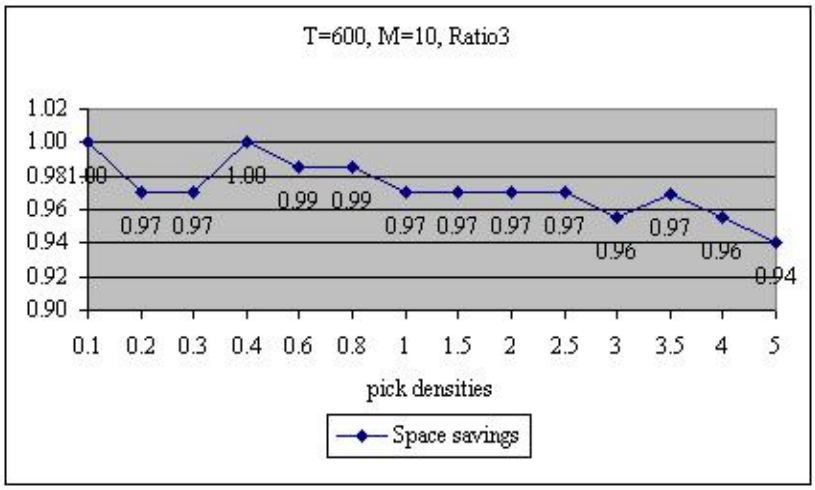

T=400, M=30, Comparison of number of cross aisles for both cases giving best saving ratios of equally spaced case
number of cross aisles
pick densities
Equally spaced
Unequally spaced

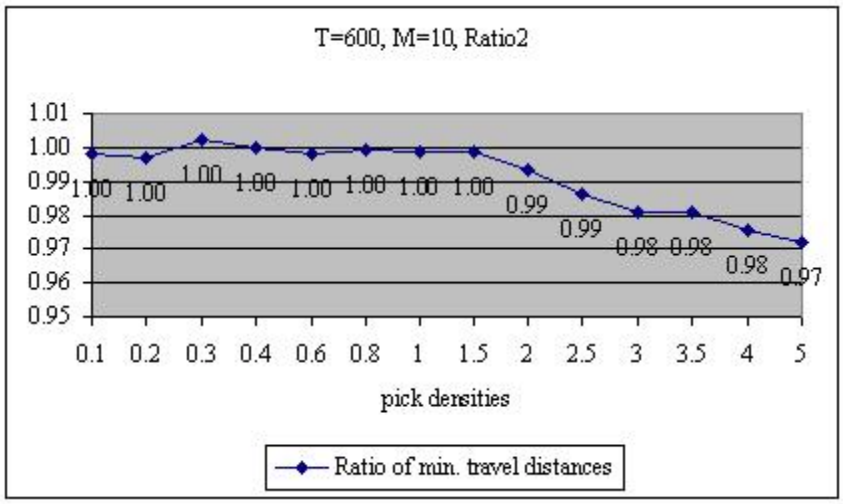

T=400, M=30, Ratio3
pick density
Space savings

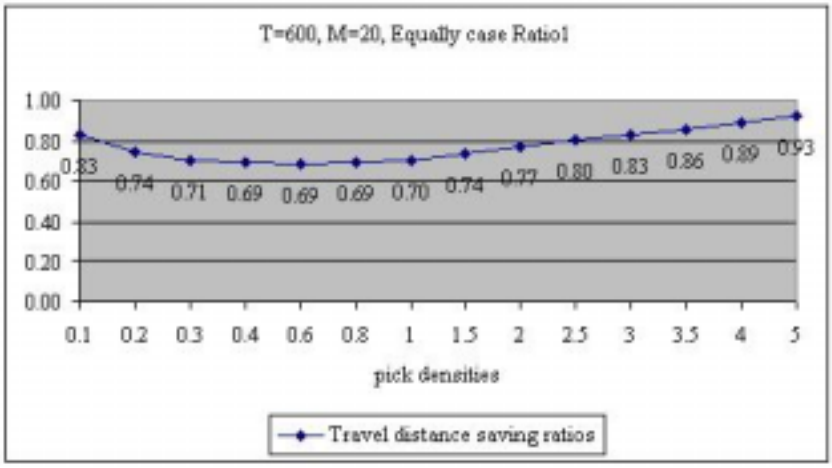

T=400, M=30, Ratio2
pick densities
Ratio of min travel distances

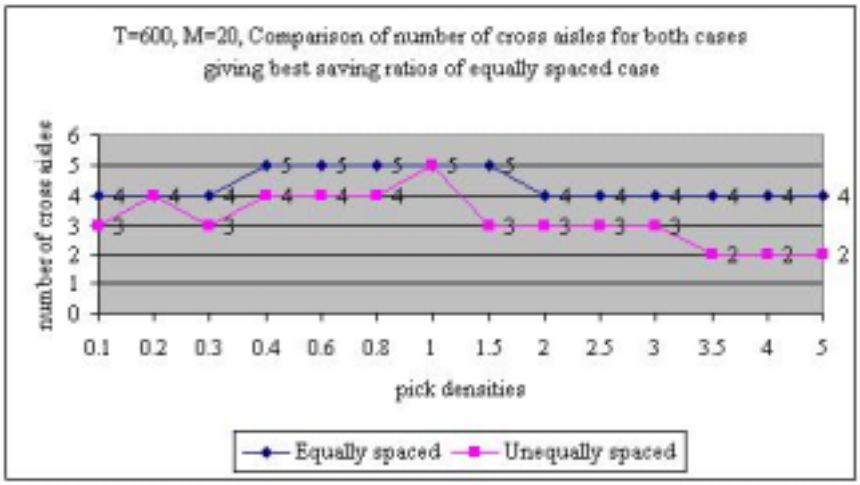

T=600, M=10, Equally case Ratio1
1.00
0.80
0.60
0.40
0.20
0.00
0.92 0.82 0.77 0.74 0.72 0.72 0.73 0.75 0.79 0.82 0.85 0.87 0.90 0.94
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance savings

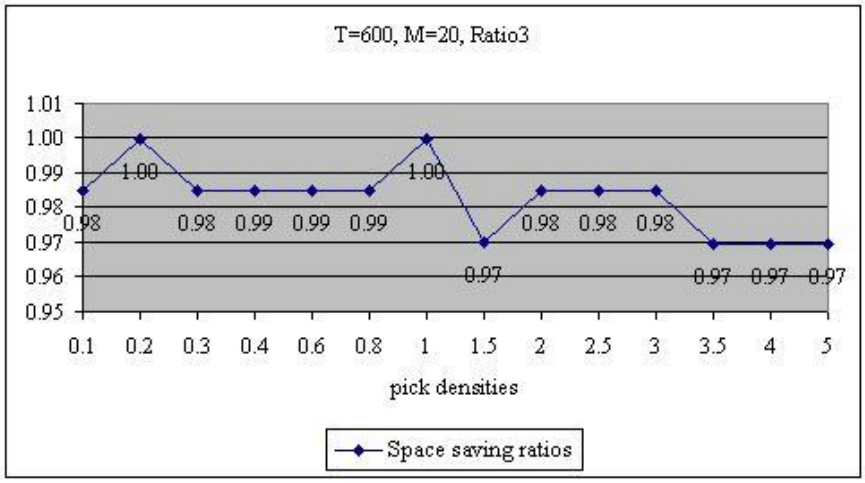

T=600, M=10, Comparison of number of cross aisles for both cases giving best saving ratios of equally spaced case
number of cross aisles
6
4
2
0
5 5 5 4 5 5 5 5 5 5 5 5 4 5 5
3 3 3 4 4 4 3 3 3 3 3 2 2 2 1
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally case
Unequally case

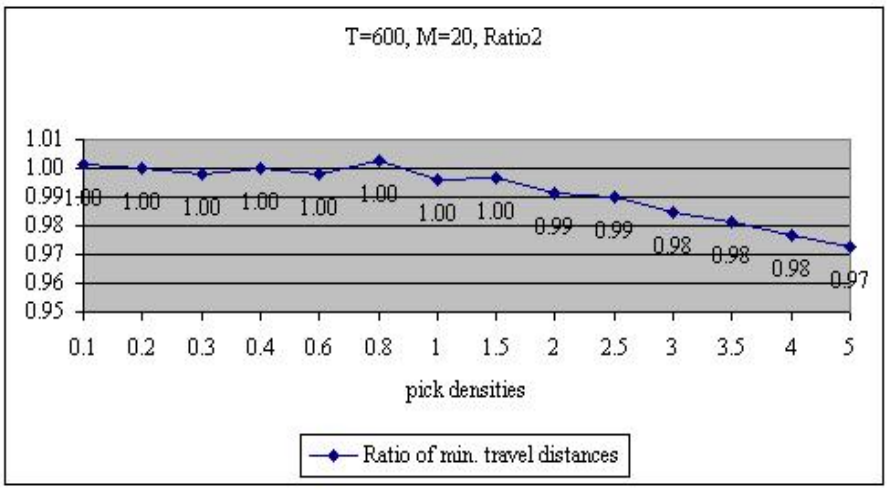

T=600, M=10, Ratio3
1.02
1.00
0.98
0.96
0.94
0.92
0.90
1.00 0.97 0.97 1.00 0.99 0.99 0.97 0.97 0.97 0.97 0.96 0.97 0.96 0.94
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space savings


T=600, M=10, Ratio2
1.01
1.00
0.99
0.98
0.97
0.96
0.95
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.99 0.99 0.98 0.98 0.98 0.97
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances


T=600, M=20, Equally case Ratio1
1.00
0.80
0.60
0.40
0.20
0.00
0.83 0.74 0.71 0.69 0.69 0.69 0.70 0.74 0.77 0.80 0.83 0.86 0.89 0.93
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Travel distance saving ratios


T=600, M=20, Comparison of number of cross aisles for both cases
giving best saving ratios of equally spaced case
number of cross aisles
6
5
4
3
2
1
0
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Equally spaced
Unequally spaced


T=600, M=20, Ratio3
1.01
1.00
0.99
0.98
0.97
0.96
0.95
0.98 1.00 0.98 0.99 0.99 0.99 1.00 0.97 0.98 0.98 0.98 0.97 0.97 0.97
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Space saving ratios


T=600, M=20, Ratio2
1.01
1.00
0.99
0.98
0.97
0.96
0.95
1.00 1.00 1.00 1.00 1.00 1.00 1.00 1.00 0.99 0.99 0.98 0.98 0.98 0.97
0.1 0.2 0.3 0.4 0.6 0.8 1 1.5 2 2.5 3 3.5 4 5
pick densities
Ratio of min. travel distances

9. REFERENCES

1. Chew, E.P., Ching T.L., 'Travel time analysis for general item location assignment in a rectangular warehouse', *European Journal of Operational Research*, 112, pp.582-597, 1999

2. Cormen, H.T, Leiserson E.C., Rivest L.R, Stein Clifford, *Introduction to Algorithms*, McGraw-Hill, 2002

3. Daniels L.R., Rummel L.J., Schantz R.,'A model for warehouse order picking', *European Journal of Operational Research*, 105, pp. 1-17, 1998

4. De Koster, R., Roodbergen, K.J., Van Voorden, R., 'Reduction of walking time in the distribution center of De Bijenkorf', *New Trends in Distribution Logistics*, Springer-Verlag Berlin Heidelberg, 1999

5. De Koster, R., Van der Poort, E. 'Routing order pickers in a warehouse: a comparison between optimal and heuristic solutions', *IIE Transactions*, 30, pp. 469-480, 1998

6. Deitel, H.M., Deitel H.M., *How to Program*, Prentice Hall, 2001

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

8. Gibson R.D., Sharp G., 'Order batching procedures', *European Journal of Operational Research*, 58, pp. 57-67, 1992

9. Goetschalckx, M., Ratliff H.D., 'Order picking in an aisle', *IIE Transactions*, 20, pp. 53-62, 1988

10. Jarvis M.J., McDowell D.E., 'Optimal product layout in an order picking warehouse', *IIE Transactions*, 23, pp.93-102, 1991

11. Petersen, C.G., ‘An evaluation of order picking routing policies’, *International Journal of Operations and Production Management*, 17, pp. 1098-1111, 1997

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

13. Roodbergen, K.J., De Koster, R., ‘Routing order pickers in a warehouse with a middle aisle’, *European Journal Of Operational Research*, 133, pp. 32-33, 2001

14. Roodbergen, K.J., De Koster, R., ’Routing methods for warehouses with multiple cross aisles’, *International Journal of Production Research*, 39, pp. 1865-1883, 2001

15. Rouwenhorst, B., Reuter B., Stockrahm, V., Van Houtum, G.J, Mantel R.J., Zijm W.H.M, ‘Warehouse design and control: framework and literature review’, *European Journal of Operational Research,* 122, pp. 515-533, 2000

16. Ruben, R.A., Jacobs F.R., ‘Batch construction heuristics and storage assignment strategies for walk/ride and pick systems’, *Management Science*, 45, pp. 575-596, 1999

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

18. Speranza, M.G., Staehly, P., *Lecture Notes in Economics and Mathematical Systems*, Springer, 2000

19. Van den Berg, J.P., Zijm, W.H.M, ‘Models for warehouse management: classification and examples’, *International Journal of Production Economics,* 59, pp. 519-528, 1999

20. Vaughan, T.S., Petersen, C.G., ‘The effect of warehouse cross aisles on order picking efficiency’, *International Journal of Production Research*, 37, pp. 881-897, 1999

21. Yoon, C.S., Sharp, G.P., ‘A structured procedure for analysis and design of order pick systems’, 28, pp. 379-389, 1996

22. Winston, H.P., Narasimhan, S., *On to Java*, Addison-Wesley, 1998

9. REFERENCES

1. Chew, E.P., Ching T.L., 'Travel time analysis for general item location assignment in a rectangular warehouse', *European Journal of Operational Research*, 112, pp.582-597, 1999

2. Cormen, H.T, Leiserson E.C., Rivest L.R, Stein Clifford, *Introduction to Algorithms*, McGraw-Hill, 2002

3. Daniels L.R., Rummel L.J., Schantz R.,'A model for warehouse order picking', *European Journal of Operational Research*, 105, pp. 1-17, 1998

4. De Koster, R., Roodbergen, K.J., Van Voorden, R., 'Reduction of walking time in the distribution center of De Bijenkorf', *New Trends in Distribution Logistics*, Springer-Verlag Berlin Heidelberg, 1999

5. De Koster, R., Van der Poort, E. 'Routing order pickers in a warehouse: a comparison between optimal and heuristic solutions', *IIE Transactions*, 30, pp. 469-480, 1998

6. Deitel, H.M., Deitel H.M., *How to Program*, Prentice Hall, 2001

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

8. Gibson R.D., Sharp G., 'Order batching procedures', *European Journal of Operational Research*, 58, pp. 57-67, 1992

9. Goetschalckx, M., Ratliff H.D., 'Order picking in an aisle', *IIE Transactions*, 20, pp. 53-62, 1988

10. Jarvis M.J., McDowell D.E., 'Optimal product layout in an order picking warehouse', *IIE Transactions*, 23, pp.93-102, 1991

11. Petersen, C.G., ‘An evaluation of order picking routing policies’, *International Journal of Operations and Production Management*, 17, pp. 1098-1111, 1997

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

13. Roodbergen, K.J., De Koster, R., ‘Routing order pickers in a warehouse with a middle aisle’, *European Journal Of Operational Research*, 133, pp. 32-33, 2001

14. Roodbergen, K.J., De Koster, R., ’Routing methods for warehouses with multiple cross aisles’, *International Journal of Production Research*, 39, pp. 1865-1883, 2001

15. Rouwenhorst, B., Reuter B., Stockrahm, V., Van Houtum, G.J, Mantel R.J., Zijm W.H.M, ‘Warehouse design and control: framework and literature review’, *European Journal of Operational Research,* 122, pp. 515-533, 2000

16. Ruben, R.A., Jacobs F.R., ‘Batch construction heuristics and storage assignment strategies for walk/ride and pick systems’, *Management Science*, 45, pp. 575-596, 1999

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

18. Speranza, M.G., Staehly, P., *Lecture Notes in Economics and Mathematical Systems*, Springer, 2000

19. Van den Berg, J.P., Zijm, W.H.M, ‘Models for warehouse management: classification and examples’, *International Journal of Production Economics,* 59, pp. 519-528, 1999

20. Vaughan, T.S., Petersen, C.G., ‘The effect of warehouse cross aisles on order picking efficiency’, *International Journal of Production Research*, 37, pp. 881-897, 1999

21. Yoon, C.S., Sharp, G.P., ‘A structured procedure for analysis and design of order pick systems’, 28, pp. 379-389, 1996

22. Winston, H.P., Narasimhan, S., *On to Java*, Addison-Wesley, 1998