Erasmus University Rotterdam

Master Thesis

# A Dynamic Storage Assignment for a replenishment warehouse

*Author:*
A.M. Sardha
324024

*Supervisor:*
R. Spliet

*Second Assessor:*
C.D. van Oosterom

November 8, 2017

ERASMUS UNIVERSITEIT ROTTERDAM

## Abstract

In a replenishment warehouse are a lot of different processes during a day. These processes should go as efficient as possible in order to limit the costs of a warehouse. Labour hours are a big part of the costs in a warehouse. The labour hours in the picking area of a warehouse depend mostly on travelling to the product locations. In this thesis a closer look will be given in the travelling distance of the pickers in the picking area.

The Dynamic Storage Assignment (DSA) is developed to reduce the travelling distance of a picker. This storage assignment policy will be compared to two other polcies, namely the Systematic Storage Assignment (SSA) and the Storage Assignment by Demand (SAD). This study shows which of the three storage assignment policies gets the best results for three different datasets. 365 days are simulated in a warehouse with arriving products and order lists that have to be picked for ten different stores. This is simulated for each of the three policies for every dataset. The results of each storage assignment policy are compared to each other.

From the results follow that the DSA is the best performing storage assignment policy of the three. Even though the DSA could have extra travel distances, because products can be relocated during their time in the warehouse. The DSA still managed to obtain the lowest travel distances in most of the instances of the datasets.

# Contents

# 1 Introduction

Customers arrive every day in the stores of H&M to buy fashion and quality for the best price that is provided in a sustainable way. The products that are sold, have to be replenished in order to meet the demand of the customers. Due to capacity limitations in the stores and the high demand for the products, there is need for replenishment almost every day. This replenishment has to be done as fast as possible to minimize the lost sales. That is why there is a new replenishment warehouse in Tiel, so all the stores in the Netherlands can be replenished within 24 hours.

The replenishment warehouse consists of several departments. When the products are followed through the replenishment warehouse, they pass the biggest departments. The first department is the Transport department, also known as In & Out. This is where the products arrive in trucks from Germany or in containers from the harbour. About 95 percent of the products arrive in carton boxes, the other five percent arrive hanging on hangers, wrapped in plastics. Where the trucks are unloaded, the Buffer department comes in. The Buffer decides whether the carton boxes go to the picking area or to the buffer area. Their first priority is to place the carton boxes in the picking area. When there is no room anymore for the carton boxes, they are placed in the buffer area. The following department would be the Picking department. This is where the orders for the stores are picked. The pick area consists of six zones. These zones consist of aisles between four level racks. There are people working in every zone to pick the products from the carton boxes. This is done by pick-to-voice, which is a fast way to pick the right number of products for a specific store. The products are directly placed in crates with the number of the store on it. When a few crates are full, the picker places them on the end of an aisle. The employees of the Transport department walk to the picking area to pick up the full crates. They sort the crates to make units of five crates for one store and load them in the trucks. The trucks take the crates to their own centre, where they sort all the crates again and load them into the right trucks that deliver them the next morning at the right stores.

The focus in this thesis lies on the other five percent of the products that do not arrive in carton boxes. They arrive hanging on hangers, wrapped in plastics. These products are more delicate and often made of premium materials, such as suits and dresses. Some of these products are easily wrinkled, so they are better kept hanging than stuffed in carton boxes. A few employees of the

Transport department unload the trucks with the hanging products. They sort products directly on product number, size and colour and hang them on racks that are placed on the rails. The rails bring the products upstairs where the picking area for the hanging clothes is. The area is divided into two sides with the rails going through the centre. Both sides have two-level racks where all the products hang and the aisles between the racks start in the centre. There are three cross-aisles, one cross-aisle in the centre and one on the end of the aisles. There are also rails on the edges of the area and there are multiple rails in front of the picking area, but those are only used by the Buffer employee so he does not stand in the way of the pickers that use the rails in the centre of the area.

An employee of the Buffer receives the products upstairs and makes sure that there is room for all the racks that the Transport department sends upstairs, by temporarily placing the racks on the rails in front of the picking area. When the trucks are unloaded, the Buffer employee places the products on the two-level racks. The area is roughly divided into six sections. The products are placed in their section wherever there is room for it. That is also why there are a lot of gaps between the products in a section. The pickers do not work yet with pick-to-voice in this area, but with printed lists. Most of the lists are orders for a single store. Some lists are orders for more than one store, because these stores only need a few products. So the employees are then able to pick for multiple stores at once. These order lists are sorted with the first product closest to the depot, followed by the second closest product etc.

An employee starts at the depot by taking an order list from the pile and estimates, based on the number of products on the list, how many hanging bags are needed for the order. The empty hanging bags are placed on racks that are placed on the rails in the centre. The hanging bags are checked if they are not broken or ripped and if they are clean, then cards are placed inside and on the hanging bags with the number of the store on it. When the employee goes in an aisle to pick the products on the order list, he leaves his racks with hanging bags on the rails in the centre and just carries the products on his arm until he gets back to his racks. The products are placed in the hanging bags and then he will move on to the next location. Every employee that has to pick upstairs walks over the whole area, section by section, to pick an order for a store. This is very time consuming since the employees have to walk a lot this way. When an order is done, the hanging bags are closed and sealed. These hanging bags are then transported back downstairs with the rails, where they are loaded in the trucks.

A lot of things are not going really efficient upstairs. There are also errors made which could lead to unnecessary mispicks. The replenishment warehouse got a daily target of 99.5 percent that has to be picked. So you do not want any mispicks, especially not if they could have been avoided. The problem with the upstairs area is, that it is a bit neglected. The main priority is downstairs where there are a lot more products and a small error there could have big consequences. But that does not make upstairs unimportant, especially not when you realize that the most expensive products are upstairs. Because of the inefficiency upstairs, there is a lot to gain for the replenishment warehouse.

The purpose of this thesis will be to gain efficiency in order picking in the hanging area of the replenishment warehouse and thereby reducing the costs by using a Dynamic Storage Assignment (DSA). The next Section gives an overview of some studies that have been done about improving efficiency in replenishment warehouses. Followed by Section 3 where the Dynamic Storage Assignment problem is formulated. In Section 4 is described how the DSA works and two other storage assignment policies are explained there. The Systematic Storage Assignment and the Storage Assignment by Demand are used to compare the DSA with. The following section describes the three different datasets that are used to test the three storage assignment policies. Section 6 gives the results of every dataset for each storage assignment policy. Finally, this thesis is concluded in Section 7.

# 2 Literature review

Following from the introduction, the objective is to improve the efficiency of picking the orders for every store. This can be done in many different ways, such as warehouse design and automation. But these improvements can be very cost intensive, so a closer look will be given in less expensive possibilities to reduce the picking time that is needed to pick all orders. Since the picking time depends mostly on travelling to the location of a product, the objective in a lot of studies is to minimize the expected travel distance (Ho and Tseng, 2006). Reducing the travel distance can be done by improving the product layout in a warehouse. Relevant articles about this possibility are discussed in Section 2.1. Order batching and routing of the pickers can also contribute to a reduction in travel distance, articles about these subjects are described in Sections 2.2 and 2.3. Finally, Section 2.4 discusses articles about warehouses with more than one level.

## 2.1 Product Layout

According to Jarvis (1984) the expected distance travelled can be minimized by taking a closer look to the product layout within a warehouse. Assume there is a depot where the picker starts picking an order and returns when the order is finished. The total distance travelled can be divided by the In-Aisle direction and the Cross-Aisle direction. By placing the products with the same demand together in one aisle, the expected distance travelled in the In-Aisle direction can be minimized. To minimize the expected distance travelled in the Cross-Aisle direction the aisle with the highest demand has to be placed closest to the depot. So the aisle with products that have the lowest demand, has to be placed farthest away from the depot. This study showed results of 40 percent decrease in mean distance travelled per order.

It could be difficult for some warehouses to base their layout on demand, especially when new products have to be stored. Another way to look at the product layout within a warehouse is considering the order or picking volume of products. Petersen (1999) compares the performance of volume-based storage to random storage in his study. Volume-based storage places high volume products closest to the depot and the lowest volume products are stored farthest away from the depot. This study uses two variations of volume-based storage, namely diagonal and within-aisle storage. With the within-aisle storage you get products with the same volume in one aisle and with the diagonal storage this can differ within one aisle. In these storage environments, various routing heuristics and an optimal routing are evaluated. The heuristics that were evaluated are composite, largest gap and transversal. The composite heuristic minimizes the travel distance between the farthest picks in two adjacent aisles. With the largest gap heuristic, the picker enters an aisle as far as the largest gap in that aisle. A gap is represented as the separation between two adjacent picks, between the front aisle and the first pick or between the back aisle and the last pick. This results in return routes from both ends of the aisle, if the largest gap is between two adjacent picks. Otherwise, the picker performs one return route from the front or the back aisle. The transversal heuristic lets a picker only enter an aisle if it contains an item to pick and exit that aisle through the other end. Also the impact of travel speed and picking rates on routing and storage policy performance is examined. All the results in this study are highly dependent on the travel speed and the picking time. The results show that volume-based storage produce significant cost savings over random storage. With within-aisle storage as the best overall volume-based storage policy regardless of aisle travel restrictions. Which backs up the study of Jarvis (1984). The routing heuristic that has the smallest average percentage gap from optimum is the composite heuristic.

### 2.1.1 Storage Assignment Policies

Some other common storage policies are random storage, class-based storage and dedicated storage. Hausman et al. (1976) compared the operating performance of these three storage assignment rules. Random storage and dedicated storage can be seen as extreme class-based storage policies. With random storage there is only one class in which all products belong. With dedicated storage every product got its own permanent location in the warehouse. That means that there are as many classes as there are products. According to the research of Hausman et al. (1976) significant potential reductions in travel times seem to be possible with class-based turnover storage assignment.

These storage policies of (Hausman et al., 1976) are applied to a stochastic environment by Thonemann and Brandeau (1998). In this paper a discrete storage rack and a continuous storage rack are considered. They developed an expression for expected one-way travel time given uniform and exponentially distributed demand. They showed that the turnover-based policy applied to the stochastic environment minimizes one-way travel time and both the turnover-based policy and class-based assignment reduce the expected storage/retrieval time in the stochastic environment compared with random assignment.

Classes can be made in different ways. Muppani and Adil (2008a) use a non-linear integer programming model for formation of storage classes. This model is solved with a branch and bound algorithm. The Class Formation and Allocation Model (CFAM) has the objective to minimize the sum of storage space cost and handling cost. They assign products with a lower cube per order index (COI) to classes closer to the depot in order to minimize the costs. COI is the ratio of the products storage space requirement to the number of requests for the product (Malmborg and Bhaskaran, 1990). The paper of Muppani and Adil (2008a) showed that there are significant savings in using class based storage policy compared to random and dedicated storage policies. This paper also showed that the branch and bound algorithm is computationally much more efficient than a baseline dynamic programming algorithm.

Another approach for class-based storage assignment is given by Muppani and Adil (2008b). This paper describes an integer programming model for class formation and storage assignment that considers all possible product combinations, storage-space cost and order-picking cost. To solve this model, a simulated annealing algorithm (SAA) is developed. This SAA performed also better than the dynamic programming algorithm for class formation with COI ordering restriction. They also observed that the variability of inventory products has an influence on which storage policy is the best. For inventories with a high variability of products, class-based storage assignment offers a greater benefit than dedication storage assignment.

### 2.1.2 Zone Picking

Zone picking can also be used to increase efficiency of picking orders. The picking area is divided in multiple zones and every employee is assigned to one zone. That means that an employee only picks products from its own picking zone. An order list of a store is now divided for the different zones, so one order for a store is picked by several employees in stead of one. This way the pickers need to traverse a smaller area. The disadvantage is that the order of one store is split and must be merged again. This problem can be coped with in different ways. The first approach is the pick-and-pass system. One employee starts with picking the order of one store in its own zone. When his part of the order is finished, he passes the products and the order list to the other picker in the next zone. An order is finished when all relevant zones are visited. Another approach for zone picking is parallel picking. A number of employees start on the same order with picking in its own zone. The partial orders are then merged after the picking (De Koster et al., 2007).

A study of Petersen (2002) shows that some considerations should be made to configure order picking zones. The configuration of a picking zone is considerably affected by the storage capacity of a zone, the number of products on an order list and the storage policy. When determining a proper zone configuration, the average order size should be taken into account as well as the number of picking zones to be used and the batch size. If the right configuration is used, then the savings in travel distance are substantial.

## 2.2 Order Batching

Batching orders can also be used to minimize the travel distance of pickers. There are several studies done about order batching and different methods are developed. Ho and Tseng (2006) studied the performance of different order batching methods in an order-picking warehouse. The order batching methods that they investigated, are build-up of one seed-order selection rule and one accompanying-order selection rule. A seed-order selection rule decides how the first order of a batch is selected from all the orders that have to be picked. An accompanying-order selection rule determines which order is added to the batch. Ho and Tseng (2006) investigated nine different seed-order selection rules and 10 accompanying-order selection rules. Ho et al. (2008) continued this study with more seed-order selection rules and accompanying-order selection rules. To be able to compare the results with the previous study of Ho and Tseng (2006), the same problem environment and route-planning were used. It turned out that the best combination is to select the Smallest Number of Picking Aisles as seed-order rule and Shortest Average Mutual-nearest-Aisle Distance as accompanying-order rule. That means that the best results are obtained by choosing the order that has the smallest number of picking aisles to visit, as the first order of a batch. The batch is filled with more orders till its capacity by choosing the order with the smallest aisle distance between the aisles in the batch and the aisles in the new order.

A more robust way to deal with order batching is described in a study of Hsu et al. (2005). They developed an approach based on genetic algorithms to deal with batching problems with any kind of batch structure and any kind of warehouse layout. The Genetic Algorithm order Batching

Method (GABM) directly minimizes the total travel distance, unlike other batching methods. The results encourage the use of genetic algorithms to develop an effective optimization method for resolving the real-world order batching problems.

## 2.3 Routing

Some researchers have been looking for methods to optimize order-picking routes. Ratliff and Rosenthal (1983) developed an algorithm that finds an optimal route for minimizing the travel distance in a rectangular warehouse with crossovers only at the end of the aisles. Since heuristics are easier to understand and they form routes that are fairly consistent in nature, there is still need for them to be studied. An evaluation of different order picking routing policies are discussed by Petersen (1997). According to his study, composite and largest gap are the best heuristics.

Another study about routing order pickers comes from Roodbergen and De Koster (2001). They considered a rectangular warehouse as well but they included a middle aisle. So instead of only a two cross-aisle, there are now three. For this problem they constructed a dynamic programming algorithm for calculating order picking tours of minimal length. This way they were able to evaluate whether or not a middle aisle could improve the efficiency of order picking. They did several simulation experiments with different warehouse sizes and order list sizes, in which they evaluated the different situations when there was a middle aisle and when there was no middle aisle. It appeared that, in the majority of the evaluated situations, the layout with a middle aisle resulted in lower average travel time than the basic layout without a middle aisle.

## 2.4 Multi-level Warehouse

All these studies do not consider warehouses with multi-level storage. There are only a few studies that took a closer look at multi-level rack distribution warehouses, such as Chan and Chan (2011). In their study they performed twenty-seven experiments by simulation on different combinations of policies, namely three storage assignment policies, three routing policies and three pick densities. The considered storage policies are random storage, horizontal and vertical ABC class-based storage. With horizontal ABC class-based storage the fast moving products (i.e. A-products) are assigned to a location close to the depot, with vertical ABC class-based storage the A-products are stored at lower levels of the racks. For assigning the products in their class, they make use of the

COI index and EIQ analysis. The EIQ analysis is about three logistics factors, namely order Entry, Item and Quantity. It is applied to analyse the orders of customers and identify which products are important. The EIQ is also described in Zhao et al. (2012). Chan and Chan (2011) focusses on three common routing heuristics in a manual-pick multi-level warehouse. These heuristics are transversal, return and composite. The return method means that an order picker enters and leaves an aisle from the same end if that aisle contains an item that has to be picked. The pick density is defined as the variety of products in a customer order which affects the performance of picking. These densities are 30%, 15% and 5%, they are chosen based on historical data. The objective is to improve picking performance in terms of travel distance and order retrieval time. Because the minimum moving distance is not the same as the minimum picking time. From the results it turns out that for multi-level rack warehouses, the use of vertical ABC class-based storage assignment policy can improve the performance of picking in terms of total order retrieval time. For a single-level rack warehouse shorter travel distance can be obtained by using the horizontal ABC class-based storage assignment policy. This study also confirms that the composite heuristic has superior performance as routing policy.

# 3 Problem Formulation

This thesis considers a rectangular warehouse that is divided by a broad middle aisle in two sides, $A$ and $B$. Both sides consist of $n$ two-level racks of the same size with aisles separating them. These aisles are broad enough to walk in both directions with multiple people without any trouble. The second level of a rack can be reached without any resource. For the ease of further formulation, some parameters should be clarified.

| | |
|---|---|
| $h$ | The number of a two-level rack in the warehouse per side with $h \in \{1,\dots,n\}$. |
| $r$ | The length of a rack in meters. |
| $g$ | The side of the warehouse with $g \in \{A, B\}$. |
| $l$ | The level of a rack with $l \in \{1, 2\}$. |
| $v$ | The section on a rack with $v \in \{1,\dots,f\}$. |
| $t$ | The number of the day in the warehouse with $t \in \{0,\dots T\}$. |
| $w$ | A unique product in the warehouse with $w \in \{1,\dots,p\}$. |
| $u_w$ | The storage space that one product $w$ needs on a rack in meters. |
| $b_{wt}$ | The size of a batch of product $w$ in number of products that arrive at day $t$. |

There are also some variables used for the formulation of the problem.

| | |
|---|---|
| $p^{gl}_{hvt}$ | The number of products that is at side $g$, on rack $h$, at level $l$ and in section $v$ at the end of day $t$. |
| $u^{gl}_{hvt}$ | The storage space that one product needs in meters, that is at side $g$, on rack $h$, at level $l$ and in section $v$ at the end of day $t$ . |

A rack is $r$ meters long and divided in $f$ sections. The sections start with counting from the middle aisle. So section 1 is closest to the middle aisle and section $f$ the furthest away. The sizes of these sections are variable since the sections are divided by cards that can be moved along the whole rack. The number of products ($p^{gl}_{hvt}$) times the storage space per product ($u^{gl}_{hvt}$) gives the sizes of the section $v$, on rack $h$, at level $l$, at side $g$ at the end of day $t$. One section contains the same $p^{gl}_{hvt}$ products, so every batch of products is in a different section. A batch of products is a number of products that consist solely of the same products. The total size of all sections on one rack should be less or equal to $r$, the size of the whole rack.

This warehouse will be used to develop a Dynamic Storage Assignment (DSA) in order to gain efficiency for order pickers. The DSA will assign locations to new products and it will relocate products that are already in the warehouse. Section 3.1 will formulate this problem. The objective of this thesis will be to minimize the distance that pickers need to travel in a replenishment warehouse to pick the order list for a store. Therefore it is necessary that the routes that the pickers will make in the warehouse are minimized, this problem will be formulated in Section 3.2.

## 3.1 Dynamic Storage Assignment

Every day products arrive in the warehouse that need to be stored, for a timespan of $T$ days. All the products come in batches of different sizes, $b_{wt}$. The product batches, that arrive at day $t$, need to be assigned to a section of a rack for storage at the end of the day. So they can be picked at the beginning of the next day. This way the storage assignment starts after the products are picked for that day. A batch can only be stored at side $g$, on rack $h$, at level $l$ if the available space on the rack is larger or equal to $b_{wt}u_w$. This means that all the sections can move on a rack, but only during storage assignment. When all the products are placed on the racks, the sections are made as small as possible and shifted as close to the middle aisle as possible. Then the locations of the sections along with their sizes will be fixed. So the next day during the picking, the locations of the products do not shift. After picking all the products from one day, some space becomes available within the sections. Then for the storage assignment, it is again possible to move the sections along the rack and change their sizes for the storage of the new batches of products. This process is repeated every day, for $T$ days.

Once a batch of products is assigned to a section, it will not necessarily stay there until the last product is picked. It could be that some products are stored in sections that are inefficient over time, i.e., orders could be picked faster if some products would be reassigned to other sections. Dynamic Storage Assignment will also relocate products in the warehouse if this is more profitable. The reshuffling will have extra costs, but it still can be profitable if it reduces the picking cost enough. These reshuffling costs are the meters that an employee needs to walk to relocate all the products in the warehouse that are assigned to a new location. Reshuffling can take place every $q$ days, with $q \in \{1,\ldots,T\}$. Consider that a period $P_y$ (with $y \in \{1,\ldots,z-1\}$) consists of $q$ days, with $z = \lceil \frac{T}{q} \rceil$. And period $P_z$ consist of $T-(z-1)q$ days. So period $P_1$ lasts from the beginning of day 1 to the end of day $q$, period $P_2$ lasts from day $q+1$ to day $2q,\ldots,$ period $P_z$

lasts from day $(z-1)q+1$ to day $T$. Every $q$ days a closer look should be taken to which products should be relocated. This will be done at the end of the day after the storage assignment normally takes place. This way the new products are also taken into account when the relocation of all the products in the warehouse are considered. For this problem, the demand of every day and the number of products that arrive every day is known.

## 3.2 Routing

The Dynamic Storage Assignment has assigned all the products to its own section $v$, on rack $h$, at level $l$ and at side $g$ on day $t$. Once all sections are fixed at their locations, the order lists can be printed so the pickers can start with picking the products. Every store has got its own order list. So a picker picks all the products that one store needs and makes the order ready for transportation to that store. Then he starts from the beginning again with another order list from another store, till all order lists are picked. This way the picker makes multiple routes through the warehouse, one for every order list. Besides the routes for picking orders, an employee makes also routes for storing new products in the warehouse and for relocating products. All the distances of these routes are needed for the evaluation of the warehouse. So every meter that an employee walks in the warehouse needs to be calculated in order to make accurate decisions.

The warehouse considered in this thesis is represented by graph $G$. All the vertices and edges of the graph can be formulated:

| | |
|---|---|
| $\ell_i$ | Is the location of the $i$-th product on an order list, with $i = 1,\ldots,s$. These $s$ locations are the sections $v$, on rack $h$, at level $l$ and at side $g$ on day $t$ that the DSA assigned to the products. |
| $ma_k$ | Is the middle aisle vertex $k$ at side $A$ with $k = 1,\ldots,\lfloor\frac{n}{2}\rfloor+1$ and $n$ the total number of racks at one side. This is the point where the picking aisle $k$ of side $A$ connects with the middle aisle. The depot from where the whole tour of an order list starts and ends is represented by vertex $m_0$. |
| $mb_k$ | Is the middle aisle vertex $k$ at side $B$ with $k = 1,\ldots,\lfloor\frac{n}{2}\rfloor+1$ where also the start and end points are for the subtours. |
| $a_k$ | Is the vertex at the end of the aisle $k$ at side $A$ with $k = 1,\ldots,\lfloor\frac{n}{2}\rfloor+1$. |
| $b_k$ | Is the vertex at the end of the aisle $k$ at side $B$ with $k = 1,\ldots,\lfloor\frac{n}{2}\rfloor+1$. |

The vertices $a_k$ are connected with $\ell_i$ or with $ma_k$ if there are no pick locations on the order list in aisle $k$, for every $k$. This also applies to vertices $b_k$, which are connected with $\ell_i$ or $mb_k$ for every $k$. The edges between $a_k$ and $ma_k$ and the edges between $b_k$ and $mb_k$ represent the aisles between the racks where the products are stored. These aisles give access to two racks, except for the first and the last aisle which can only be used to reach the first and last rack respectively. This applies for an even number of racks. When the number of racks are uneven, the last aisle can reach the last rack and the second to last rack. The length of these aisles are the same $x_{aisle}$ meters. A picker can only change from aisles at one of the four vertices by using edges that connect $vert_k$ with $vert_{k+1}$ or with $vert_{k-1}$, with $vert \in \{ma, mb, a, b\}$. These edges have the same length of $x_{change}$ meters. On the middle aisle are the vertices $ma_k$ and $mb_k$ connected for every $k$, these edges are used to switch sides in the warehouse. The length of these edges are the same $x_{mid}$ meters for every $k$.

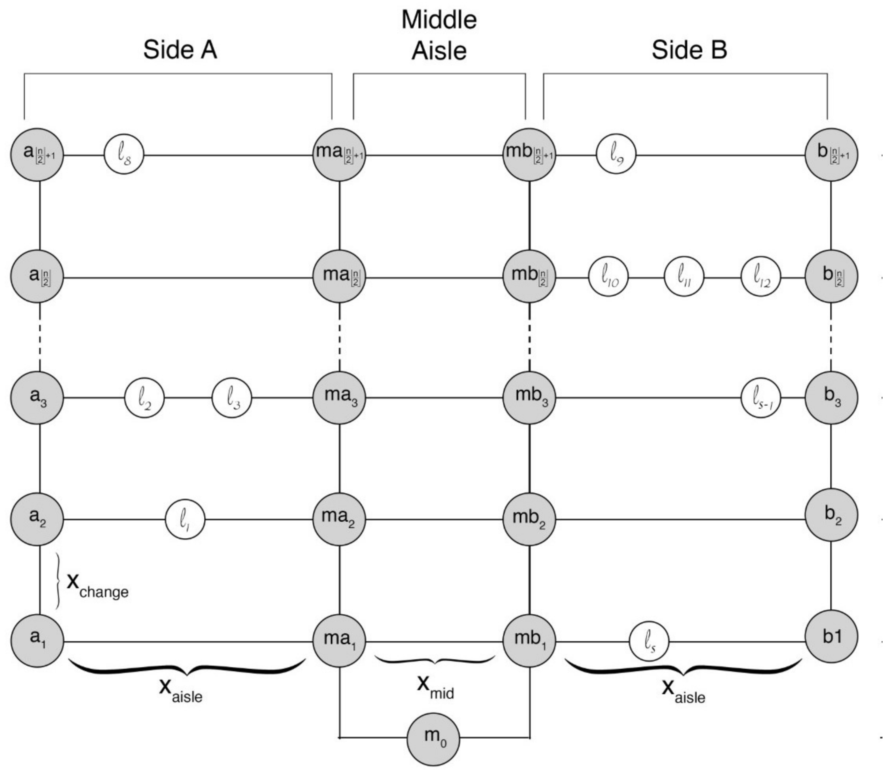

Side A
Middle Aisle
Side B
$a_{\lceil \frac{n}{2} \rceil+1}$
$ma_{\lceil \frac{n}{2} \rceil+1}$
$mb_{\lceil \frac{n}{2} \rceil+1}$
$b_{\lceil \frac{n}{2} \rceil+1}$
$\ell_8$
$\ell_9$
$a_{\lceil \frac{n}{2} \rceil}$
$ma_{\lceil \frac{n}{2} \rceil}$
$mb_{\lceil \frac{n}{2} \rceil}$
$b_{\lceil \frac{n}{2} \rceil}$
$\ell_{10}$
$\ell_{11}$
$\ell_{12}$
$a_3$
$\ell_2$
$\ell_3$
$ma_3$
$mb_3$
$\ell_{s-1}$
$b_3$
$a_2$
$\ell_1$
$ma_2$
$mb_2$
$b_2$
$x_{change}$
$a_1$
$ma_1$
$mb_1$
$\ell_s$
b1
$x_{aisle}$
$x_{mid}$
$x_{aisle}$
$m_0$

Figure 1: A representation of the warehouse as graph $G$ with $s$ pick locations.

On an order list are the locations $\ell_i$ of the products that a picker needs to visit at least once. These pick locations are somewhere in an aisle. The second level locations can be reached without any extra effort, so there are no extra costs for those locations. When a picker goes from one pick location to another pick location, he can choose to go via the middle aisle nodes ($ma_k$, $mb_k$) or the side aisle nodes ($a_k$, $b_k$). This possibility can reduce the total distance that a picker needs to walk to visit all the pick locations on an order list.

# 4 Methods

The objective of this thesis is to minimize the distance that pickers need to travel in a replenishment warehouse. There are several ways to gain efficiency in order picking in a rectangular warehouse with three cross-aisles. This thesis will look at how Dynamic Storage Assignment (DSA) can contribute to a more efficient replenishment warehouse compared to a systematic storage method and storage by demand. For this research also some routing is necessary, but the focus will lie on DSA. The storage assignments start all three with empty warehouses. This way the different policies can fairly be compared.

The different storage assignment policies will be explained in this chapter. Starting with the Systematic Storage Assignment in Section 4.1. In the same section is the Storage Assignment by Demand described. Section 4.2 explains the Dynamic Storage Assignment. The DSA uses the methaheuristic genetic algorithm. In Section 4.3 is explained how the genetic algorithm is used for storage assignment of products. The routes that pickers need to make in the warehouses are used to evaluate the storage assignments. Therefore, a nearest neighbour heuristic is used which is described in Section 4.4.

## 4.1 Storage Assignment Policies

There are different kind of storage policies that a warehouse can use. In this thesis the systematic storage assignment and the storage assignment by demand are compared to the Dynamic Storage Assignment. Every policy assigns new arrived batches of products to a location in a warehouse. If a batch of products can not be placed on one rack because the size of the batch is greater than the length of a rack, then the batch of products is cut in two. There are now two new batches of the same products instead of one. One batch will have the same size as a rack and the rest of the products are in the other batch. This applies to all three storage policies. The DSA is explained after this section, the other two policies are discussed here.

**Systematic Storage Assignment**

The Systematic Storage Assignment (SSA) is currently used in the replenishment warehouse, with this method are all products systematically stored. This means that products in the warehouse are systematically stored from the front of the warehouse towards the back. There is no need to look at the demand here, the products are just placed where there is room for it. The only idea is to fill the first racks as much as possible, then the second racks, the third racks etcetera, until all products are placed.

When new products arrive, a batch of products is randomly picked to be stored first. The systematic storage assignment first checks if there is place on the first rack, at the first level and at side $A$ for the batch of products. If there is no room there the second level of the first rack at side $A$ is checked. Then side $B$ is checked at rack 1 at level 1, followed by level 2. If there is still no room for the batch of products the second racks are checked in the same order. This continues till a rack with enough room is found. All the products on this rack are shifted to the middle aisle, to make room for the new batch of products that will be stored on the end of the rack. Then another batch of new arrived products is randomly chosen to be stored. It starts again by checking the first racks, then the second racks etcetera. This is repeated again until all products are placed.

This storage assignment policy is evaluated by calculating the distance that the pickers need to travel to pick all the products from the order lists. Together with the distances to store all arrived products at a location, gives a total distance in meters. This distance can be compared with the other storage assignment policies. Section 4.4 gives a more detailed explanation of the order lists and the routes that the pickers will travel.

**Storage Assignment by Demand**

This storage policy is actually the same as is described in Section 4.2 for the Dynamic Storage Assignment. The only difference is that storage assignment by demand is not dynamic. So products do not have the possibility to be relocated, thus the genetic algorithm is also not used for the storage assignment by demand. This storage assignment policy is evaluated in the same way as the SSA by calculating all the distances that need to be travelled in order to pick all the products from the order lists and to store all arrived products.

## 4.2 Dynamic Storage Assignment

The products that arrive in the warehouse have to be stored in section $v$, on rack $h$, at level $l$, at side $g$. There are different storage policies that can be used in order to minimize the travel time of the pickers. The Dynamic Storage Assignment will be based on the demand of the products. Since products with a high demand will be picked more often, it will be convenient if the distance from the depot to those products is small. This will minimize the travel time of the pickers. The demand of the arriving products for every day is known. With this data, the products that arrived at day $t$ can be ranked from highest demand to lowest demand. A product with a high demand will get a section close to the depot and a product with a lower demand will get a section further away from the depot. For ranking the products, only the demand of the upcoming $q$ days are used. Since the products are fashion items, the demand can strongly depend on the season of the year.

**Arrived Products**

The demand of the products in the warehouse at day $t$ have to be considered when finding locations for a new arrived batch of products. The DSA starts with the arrived batch of products $w$ that has the highest demand. This way the arrived batches of products are sorted from 1 (the batch of products with the highest demand) to $B_t$, with $B_t$ being the number of arrived batches of products on day $t$. Day $t$ has $B_t + 1$ moments for which the number of products in every section in the warehouse are updated. These moments are denoted with parameter $n \in \{0, \ldots, B_t\}$. The first moment ($n = 0$) is after the last product is picked for the day and before the first product is assigned to a location in the warehouse. Every time a product is picked, that section reduces in size. So the number of products of every section has to be updated before the storage assignment begins. Then the arrived batch of products with the highest demand will be assigned to a location, this is the second moment ($n = 1$) of day $t$. First the DSA checks if it is a replenishment of a product that already has a location in the warehouse. If that is the case, the DSA tries to place the arrived products in the same section. If this would violate condition 4.1 than it is treated as a new arrived product and another location will be searched. DSA considers every rack $h$, at every level $l$, at both sides $g$ for potential storage of the new products if the condition 4.1 holds. Starting with the racks closest to the depot.

$$
\begin{aligned}
&\sum_{v} p^{gl}_{hvt_{n-1}} u^{gl}_{hvt} + b_{wt} u_w \leq r \qquad , \text{with}\\
&\quad p^{gl}_{hvt_0} = p^{gl}_{hv(t-1)} - \text{picked products from rack } h\text{, level } l\text{, side } g \text{ and section } v \text{ at day } t\\
&\quad p^{gl}_{hvt_n} = \begin{cases} p^{gl}_{hvt_{n-1}} + b_{wt} & \text{, if product } w \text{ is placed in rack } h\text{, level } l\text{, side } g \text{ and section } v \text{ at day } t \\ p^{gl}_{hvt_{n-1}} & \text{, otherwise} \end{cases}
\end{aligned}
\tag{4.1}
$$

When the condition 4.1 holds for a certain rack, it means that there is enough room on that rack for the new arrived batch of product $w$. Now the demand of all the batches of products on that rack are compared to the demand of the new product, to determine its place on that rack. If the demand is higher than the new batch of products, then the product remains at its location in the same section. If the demand is lower than the new batch of products, then the product will shift on the rack, with the size of the new batch of products, further away from the depot and its current section number increases with one. This way the new batch of products gets a section on that rack if it would be placed there. This potential location for the new arrived batch of product is saved. In the same way every rack in the warehouse is systematically considered as potential storage location until the new arrived batch of products gets a potential location with section 1 on a certain rack. The first section on a rack is the closest to the depot, so any other rack that comes after the rack with potential storage section 1, will give a potential storage location that is as close to or further away from the depot. From all these potential storage locations the one with the shortest distance to the depot will be chosen. This is repeated and $n$ is increased with 1 until all new arrived batches of products are assigned to a location and $n = B_t$. Note that at the end of the day $p^{gl}_{hvt_{B_t}} = p^{gl}_{hvt}$.

This storage assignment comes with costs that should be taken into account. These costs are the walking distances to store the products at its assigned location. When the arrived products are unloaded from the trucks at the replenishment warehouse, they are temporarily placed on racks in front of the picking area (as is described in Section 1). Since these products are unloaded in an unknown order, it is difficult to say where they are temporarily placed. So it is assumed that the starting point for the storage of new arrived products is at the depot.

Since the demand of the products can be influenced by the seasons of the year, it can change a lot. This means that a product can have a high demand when it arrives in the warehouse, but in the next season a very low demand or the other way around. Due to this fluctuation in demand, it can be profitable to relocate some products in the warehouse. So at the end of a period $P_y$ the reshuffling of products can take place. The DSA uses a genetic algorithm for this reshuffling, so the products will have a more efficient location for the upcoming period $P_{y+1}$.

## 4.3 Genetic Algorithm

The genetic algorithm is a metaheuristic that is inspired by nature. It uses the principals of the theory of evolution and natural selection. Therefore, the genetic algorithm creates a population with individuals. An individual represents a warehouse with all the products stored at a certain location at the beginning of every period. Every individual has the same products in the same warehouse but at different locations, this makes every individual unique. In nature the information of an individual, is stored in DNA. This DNA consist of chromosomes that has all the information of an individual that makes him unique. The genetic algorithm also uses a metaphorical chromosome to store the information of the locations of every product in a warehouse. To make a chromosome from the replenishment warehouse, every product in the warehouse needs to be numbered with a product ID from 1 to $p$. The product ID of every product is connected to the location of that product and some information. This product information consist of the number of products, the width per product, the section size and its demand. This way a matrix of the products can be made with their product ID, corresponding storage location and information next to it. See Table 1 for an example of a schematic representation of one layer of a chromosome. This gives the first layer of the chromosome that represents the warehouse at the beginning of the first period. During this period new products arrive, some current products are replenished and some products leave the warehouse. So the next period starts with another variety of products. These products with their locations are in the second layer of the chromosome. This way the chromosome is build of $z$ layers, each layer represents the warehouse at the beginning of a period. See Figure 2 for a schematic representation of a whole chromosome with $z$ layers.

| **ProductID** | **Product location** | | | | **Product information** | | | |
|---|---|---|---|---|---|---|---|---|
| | Rack | Side | Level | Section | number of products | width per product | section size | demand |
| 1 | 1 | 2 | 1 | 4 | 20 | 0.015 | 0.3 | 50 |
| 2 | 1 | 1 | 2 | 2 | 15 | 0.04 | 0.6 | 32 |
| 3 | 4 | 2 | 1 | 1 | 36 | 0.015 | 0.54 | 45 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $p$ | $h$ | $g$ | $l$ | $v$ | $p^{gl}_{hvt}$ | $u^{gl}_{hvt}$ | $p^{gl}_{hvt} \times u^{gl}_{hvt}$ | $d_p$ |

Table 1: An example of a schematic representation of one layer of a chromosome.

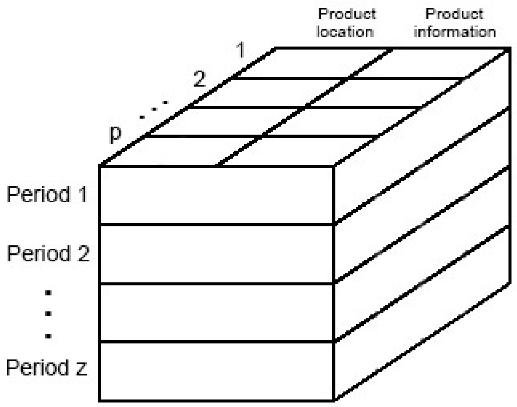


Figure 2: A schematic representation of a whole chromosome. It is build up from $z$ layers, each layer represents the warehouse at the beginning of a period.

### 4.3.1 Initialisation

First an initial population of $2 \times Pop$ chromosomes has to be created to start with the genetic algorithm. The first layer of the first chromosome is the current replenishment warehouse. Since all storage policies start with an empty warehouse, the first layer will be empty. With this warehouse the whole first period is simulated to obtain the warehouse with new arrived products and without the picked products from the first period. This creates the warehouse at the beginning of the second period, thus the second layer of the chromosome. In the same way are the following $z - 2$ layers created to complete the first chromosome.

The other $2Pop - 1$ chromosomes are generated with all the products assigned to random locations in the warehouse. Randomisation is important for the genetic algorithm in order to investigate a big part of the solution space. That is why the initial population is filled up with these random chromosomes. A random chromosome is generated per layer from the first chromosome.

All the products from the second period are removed from their current locations and are placed in a pool (the first layer of all chromosomes are empty). Then randomly one batch of products $w$ will be picked from the pool and assigned to a location in a similar way as is done with arrived products (see Section 4.2 Arrived Products). Only the demand of the products is not considered here due to the priority of randomisation. The location will be systematically assigned by first trying to place the batch of products $w$ on rack $h = 1$, at level $l = 1$, at side $g = A$. If the condition 4.1 holds, then the batch of products $w$ will get the next available section $v$ on that rack as potential location. Then level 2 of the same rack is checked for enough room to place the new batch of products, followed by the other side of the warehouse (side $B$). There on rack 1 and level 1 is their potential storage location of product $w$ checked. Subsequently comes level 2 of side $B$ and then the next rack. This continues systematically until a potential location with section 1 is found. Then from all these potential locations, the one closest to the depot is chosen. Then the next batch of products get randomly picked from the pool to be assigned in the same way. This continues until all products are assigned to a location to complete the warehouse in the beginning of the second period and thus the second layer of a chromosome. The layer is then sorted by the product ID, starting with the product with the lowest ID number. Products with the same ID but at different locations are sorted by the number of products at that location, with the lowest number of products on top. If the chromosome layers are not sorted, it could happen that the reproduction creates double locations. This is avoided by sorting the chromosomes. This procedure is repeated for every period to complete a whole chromosome.

Once a chromosome is completed, it follows a small training to improve itself. The products on one rack of the warehouse, that are placed there randomly, are now sorted by their demand. The product on the rack with the highest demand will get section 1, the product on the rack with the second highest demand will get section 2 etcetera. So the product with the highest demand of that rack gets a location next to the middle aisle and the product with the lowest demand a location furthest away from the middle aisle. This is done for every rack in the warehouse and for every layer of the chromosome.

### 4.3.2 Selection

Once the initial population consist of $2 \times Pop$ chromosomes, the fitness level ($f_c$) of every chromosome $c$ will be calculated by formula 4.2.

$$f_c = (Dist_c + Pen_c)^{-1} \times 10^7 \tag{4.2}$$

This fitness level of chromosome $c$ consists of the distance travelled by the employees and a penalty. $Dist_c$ is the distance that the pickers will travel in the warehouse to pick all the orders from every period and the distance that is needed to store all the arrived products at their assigned locations in meters. The distance of the pickers will be calculated with a nearest neighbour heuristic as described in Section 4.4.1.

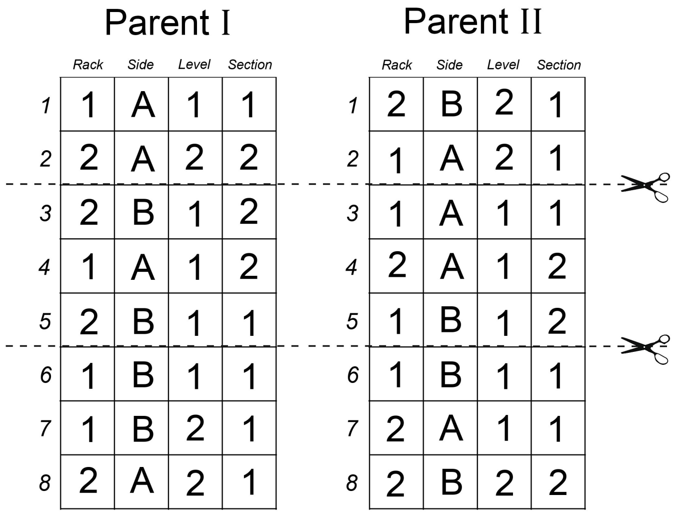
The penalty ($Pen_c$) can be given for every period in meters. It could be that at the end of a period $P_y$ the product locations differs from the locations in the chromosome layer $y + 1$. This means that those products are relocated to the locations of the chromosome layer $y + 1$. Therefore a reshuffle list will be made with the old and new locations of the products that are relocated. This relocation of products will be done as described in Section 4.4.2. The distance that is necessary to complete this reshuffle list will be the penalty for the fitness level. These fitness levels are first used to sort all the chromosomes, so the chromosome with the highest fitness value will be chromosome 1. The bottom half of the chromosome list will be removed so the best $Pop$ chromosomes can only be selected as parents for the reproduction of the next population. To create the next population, two different parents from the best $Pop$ chromosomes are selected to create an offspring. A parent $c$ (with $c \in \{1, \ldots, Pop\}$) is selected with the probability $Prob_c$ that is calculated with formula 4.3. The higher the fitness level of chromosome $c$, the higher the probability is to be chosen.

$$Prob_c = \frac{f_c}{\sum\limits_c f_c}\ , \qquad \text{with } \sum_c Prob_c = 1 \tag{4.3}$$

### 4.3.3 Reproduction

The reproduction of a chromosome is also done per layer, starting with the second period (second layer) since the first layer is empty. For the reproduction a two-point crossover method is used.

There are randomly two points chosen in the strings of the parent chromosomes of the same layer. These points cut the chromosome layers, each at the same place, in three parts. Then randomly one of the two chromosomes are chosen for each part to create the first layer of the offspring, without choosing all three parts from the same chromosome. So there are six different possibilities to make an offspring. This is done for every layer to complete the chromosome of the offspring. See Figure 3 for a simplified schematic representation of the reproduction of two parent chromosomes $I$ and $II$ of one layer of a chromosome. For simplicity is the production information left behind. Figure 4 gives an overview of the six different possibilities to make an offspring.



Parent I

| | Rack | Side | Level | Section |
|---|---|---|---|---|
| 1 | 1 | A | 1 | 1 |
| 2 | 2 | A | 2 | 2 |
| 3 | 2 | B | 1 | 2 |
| 4 | 1 | A | 1 | 2 |
| 5 | 2 | B | 1 | 1 |
| 6 | 1 | B | 1 | 1 |
| 7 | 1 | B | 2 | 1 |
| 8 | 2 | A | 2 | 1 |

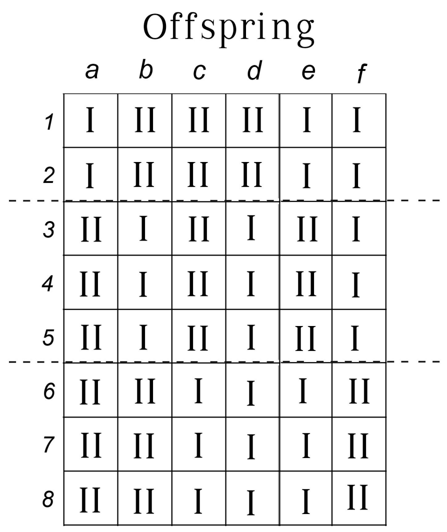
Parent II

| | Rack | Side | Level | Section |
|---|---|---|---|---|
| 1 | 2 | B | 2 | 1 |
| 2 | 1 | A | 2 | 1 |
| 3 | 1 | A | 1 | 1 |
| 4 | 2 | A | 1 | 2 |
| 5 | 1 | B | 1 | 2 |
| 6 | 1 | B | 1 | 1 |
| 7 | 2 | A | 1 | 1 |
| 8 | 2 | B | 2 | 2 |

Figure 3: An example of a simplified schematic representation of one layer of two parent chromosomes, who are selected for reproduction. They are cut at two points at the same place, after product 2 and after product 5.

These offsprings can also have mutations in every layer. With probability $\mu$ three random products of a chromosome layer will be temporally removed and with probability $2 \times \mu$ one random product of a chromosome layer will be temporally removed. Now the chromosome has to be feasible, therefore a correction mechanism is created.

The correction mechanism makes sure that the new chromosomes are feasible. It could happen that the reproduction creates an offspring with products in the same location, with sections that overlap or condition 4.1 does not hold any more. First the section numbers are checked for every rack in the warehouse. The product with the highest demand on a certain rack gets section number one, the product with the second highest demand on that rack gets section number two and so on. Then the products are shifted on that rack so that section one starts at the middle aisle and all



Offspring

|   | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| 1 | I | II | II | II | I | I |
| 2 | I | II | II | II | I | I |
| 3 | II | I | II | I | II | I |
| 4 | II | I | II | I | II | I |
| 5 | II | I | II | I | II | I |
| 6 | II | II | I | I | I | II |
| 7 | II | II | I | I | I | II |
| 8 | II | II | I | I | I | II |

Figure 4: An example of a schematic overview of the six possibilities for making an offspring from two parent chromosome layers, parent I and parent II. In this example the parent chromosome layers were cut after product 2 and after product 5 (See Figure 3).

products on the rack are shifted as close to the middle aisle as possible. This fixes any overlap on a rack and it makes the warehouse more efficient by placing the products with a higher demand closer to the depot. Now it is possible that the racks have an overflow and condition 4.1 is violated. The products at the end of the rack that cause this violation are temporally removed from the warehouse to make condition 4.1 hold again.

The product batches that the correction mechanism and the mutation has removed are ranked by their demand and treated as they were new arrived products to give them a new location (see Section 4.2 Arrived Products). Finally the chromosomes are sorted again by their product in the same way as is done in Section 4.3.1.

### 4.3.4 Evolution

Now that the chromosomes of the offsprings are feasible, their fitness levels will be calculated. The reproduction of two parents from the initial population is repeated for *Pop* times to create a new population of $2Pop$ chromosomes. From the $2Pop$ chromosomes only the half with the highest fitness levels will survive to the next generation, the other half will be deleted. Now there are again *Pop* chromosomes as the new population. From this population, the new parents will be selected and the procedure will be repeated again. This continues until the best chromosome has not changed for *Stop* generations.

## 4.4 Routing

The objective of the Dynamic Storage Assignment is to minimize the distance that pickers travel in a replenishment warehouse. Therefore, it is necessary to determine the route that pickers need to travel to pick all the products from an order list. The focus of this thesis lies with the DSA and not with the routing. The routing is only used to fairly compare the warehouses with the same products but at different locations. This way it can be determined which locations for which products minimize the distance that pickers need to travel. It is also used to compare the different storage assignment policies with each other. A nearest neighbour heuristic will be used for the routing of the pickers. This is a very intuitive way and a realistic representation of how the pickers will pick their orders.

### 4.4.1 Nearest Neighbour Heuristic

Every order list is done separately by one picker, so there is a route for every order list. On this order list are the locations of every product that needs to be picked and the number of products to be picked for one store. A picker starts every order list at the depot and goes to the nearest pick location. The picker always goes to the nearest pick location that is not visited yet. So every pick location will only be visited once. When all pick location are visited, the picker goes back to the depot to complete the route.

Every route of an order list starts at the depot. The distance from the depot to every pick location on the order list is calculated in two different ways. A pick location is actually not one point in the warehouse but it is a section with a begin point and end point on a certain rack. So the pick location can be approached from the left side of the section or the right side of the section. This gives two different distances for one location. The shortest distance is saved for every location. This creates a list of distances between the depot and every pick location on the order list. The pick location with the shortest distance is chosen as next location and removed from the order list. The section of the location becomes smaller from the left side or the right side, depending from which side the section is approached. The number of products that are picked times the width of one product is the total width that the section is reduced with. The new start location depends on the side from where the pick location was approach. Now the distances from this pick location to every other pick location on the order list is calculated in the same way. The pick location with the shortest distance is the next location to be visited and that location is then also removed from the order list. This process is repeated until the last location is visited.

It is possible that there are more than one location to pick a product. In that case, the location with the least number of products will be visited first. This is the fastest way to reduce the number of different locations for the same product. When a picker needs to pick more products than there are at one location, then that location will be made empty and a new location is added to the order list with the rest of the products that could not be picked.

When all products of an order list are picked, the picker returns to the depot with the shortest distance. All distances are added together to give the total distance for the order list.

### 4.4.2 Reshuffle list

Besides order lists that need routes to be walked, are reshuffle lists. On these reshuffle lists are all the products that are relocated by the Dynamic Storage Assignment with their current locations and the locations were they are relocated to. There is one reshuffle list for every time the warehouse gets reshuffled. An employee has to walk this reshuffle list to actually relocate the products in the warehouse. For the routing of this list, a greedy heuristic is used. The batches of products that are relocated are in different sizes. So it can be productive to pick-up a few small batches of products and then deliver them at their new locations, instead of one by one. But it could be more convenient to deliver a batch of products at its new location right after it is picked-up, because of its big size. So there is a threshold in number of products to decide whether or not another batch of products is picked-up before delivering a batch of products at its new location. This threshold will be the relocate capacity $ReCap$. There is also a variable that counts the number of products currently being transported $Tran$.

The heuristic starts at the depot with $Tran = 0$ and let the employee go to the nearest pick-up location. The routing goes in the same way as in Section 4.4.1, but now the whole section is picked empty instead of a few products. The number of products at that location $p_{hvt}^{gl}$ is added to $Tran$. If $Tran$ is greater or equal than $ReCap$, then the batch of products will be directly delivered to their new locations. If $Tran$ is smaller than $ReCap$, then the distances to the corresponding delivery location and the other pick-up locations are compared. The location with the shortest distance is visited next. If this is a pick-up location, then the number of products at that location are added up to $Tran$ and the threshold is checked. If this is a delivery location of one of the batch of products currently in $Tran$, then the products are delivered at their new location and $Tran$ is updated. This continues until all products on the reshuffle list are relocated.

A warehouse has the possibility to have more than one location for a product. The first chromosome that is produced by the DSA determines how many locations a product has in every period. That will be the same for every chromosome since the population is made from this first chromosome. The number of different locations needs to remain the same for every chromosome, because it is essential for the reproduction that the chromosomes have the same size in every period. Due to these double locations it is possible that at the end of a period of a certain chromosome there will be less or more locations than at the beginning of the next period. This is because arrived products can be stored at an existing location while that was not possible in the first chromosome. So in the first chromosome an extra location was created. This works the other way around as well. These differences in number of locations between the end of an period and the beginning of the next period, should be taken into account. With reshuffling of the products an extra location can be made or locations can be merged, so the chromosomes can be used for the reproduction again. This extends the reshuffle list and adds extra meters to the total distance that an employee travels in the warehouse to complete the reshuffle list. This distance is used as penalty in the fitness level from the genetic algorithm of Section 4.3.

# 5 Data

The three storage assignment policies described in Section 4 will be evaluated with three generated datasets. These datasets consist of two kind of products, seasonal products and all year round products. Each dataset has the same size for the replenishment warehouse. The size of the warehouse does not form any limitation for the storage policies with the three datasets. The timespan of all datasets is 365 days. The parameters for the Dynamic Storage Assignment are also the same for each dataset. The order lists and product information are in each dataset obtained in the same way, so these are only discussed in Section 5.1.

Since randomisation is involved when generating the three datasets, all three datasets are produced three times for more reliable evaluations of the storage assignment policies. Each dataset is described in this chapter, in its own section.

## 5.1 Dataset 1

The first dataset consist of 200 different seasonal products. A seasonal product is on average 6 weeks in the warehouse. So the number of days that a product is in the warehouse is normally distributed with a mean of 42 days and a standard deviation of 5 days. All numbers that are drawn from the distribution are rounded to their closest integer. The numbers that are less or equal to zero are set to one day. So all products are at least one day in the warehouse. The day that a product arrive is randomly chosen between the first day and the 365th day.

### 5.1.1 Demand

First the demand per product per day is created. The products will have a high demand in a short time. The demand of a product from its first day till its last day in the warehouse is also normally distributed with 500 products as average and a standard deviation of 100 products. Dividing this number with the number of days gives an average of the demand of a product per day. To vary the demand per day, another normal distribution is used with that average and a standard deviation of 5 products. All these numbers are rounded and negative values are set equal to zero. So the demand of all products are positive integers or zero.

There will be no demand on the first day of the year. Products arrive for the first time the day before the first demand and will be stored at the end of the day, so the demand of a product can start earliest on the second day.

Since the day that a product arrive in the warehouse is randomly chosen between day 1 and 365 and products are on average 42 days in the warehouse, it could happen that some products exceed the timespan of a year. This is corrected by removing all the demand after day 365. This results in a lower average demand per product over the whole year of 479.5 products and a lower average of days that a product is in the warehouse of 41.1 days. The different demand average is also influenced by the distribution of the demand per day.

Despite the 200 different products in this dataset, there are only 22.5 different products on average in the warehouse at once. This is due to the short period of time that a product spend in the warehouse. In the warehouse are at most 34 different products. Figure 5 gives an overview of the number of different products in the warehouse every day.

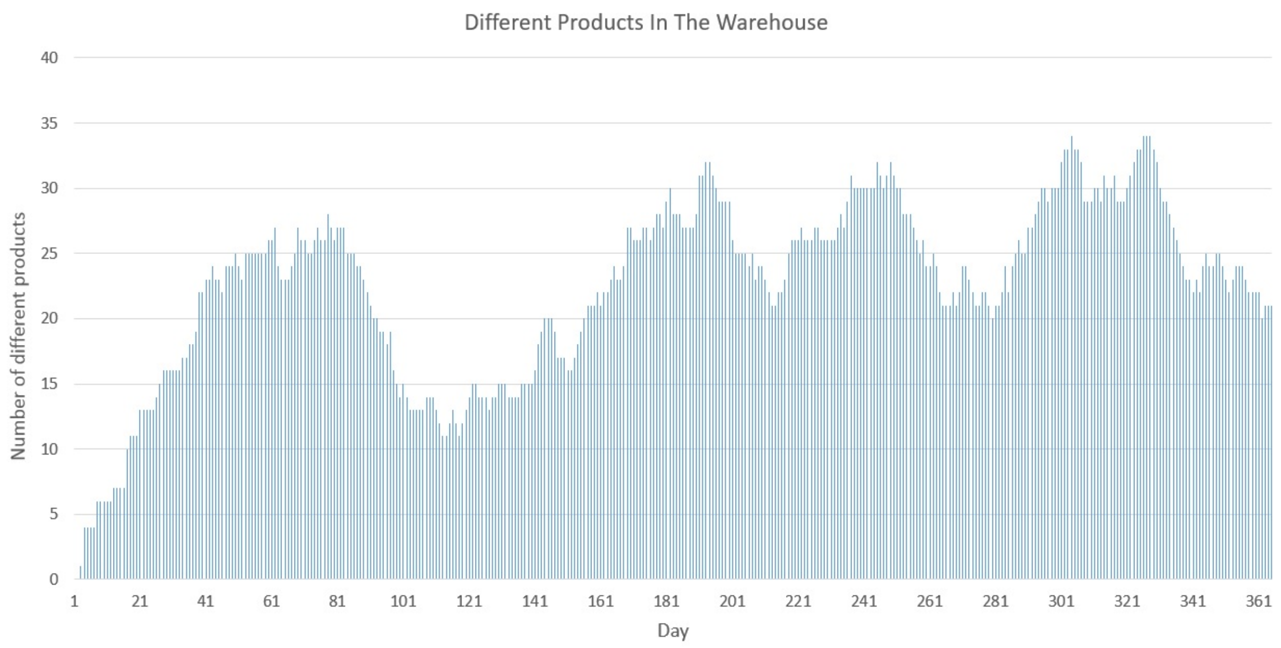


Figure 5: The number of different products in the warehouse for every day in the timespan of dataset 1.

### 5.1.2 Product Arrival

There have to be enough products in the warehouse to meet the demand per day for every product. Products are delivered to the warehouse in batches, so one product is not replenished every day.

The first delivery of a product is the day before the first demand. That way the arrived product batch is assigned to a location before the picking starts the next morning. The number of times that a product is replenished is determined by the size of the demand of the product over the whole year.

If the demand of one product is lower or equal than 250 products, then the product is not replenished. In that case, all products are delivered at once on the day before the first demand.

When the demand is between 250 and 501 products, then there will be one replenishment. Since it is not likely that the replenishment happens soon after the first day of demand and not just before the last day of the demand, it will be somewhere in between. Therefore, the replenishment day is randomly chosen after the product spend 30 percent of its time in the warehouse and before it spend 70 percent of its time in the warehouse. The amount of products in the first delivery equals the demand of the first day up to and including the replenishment day and a random number of extra products between zero and the remaining demand of that product.

When the demand of a product is more than 500 over the whole year, the product will get two replenishment days. The first delivery is also the day before the first demand. To determine the replenishment days, the period of time that the product spend in the warehouse is cut in half. The first replenishment will then happen after 30 percent of the time spend in the first half of the period and before 70 percent of the time spend in the first half of the period. The second replenishment will happen after spending 30 percent of the time in the second half of the period and before spending 70 percent of the time in the second half of the period. Now there are three moments of delivery of one product, this divides the period of demand in three periods and the demand itself in three parts. The first delivery will have an amount equal to the first period of demand plus a random part of the second period of demand. The second delivery (first replenishment) will have an amount equal to the remaining part of the second period plus a random part of the third period. The remaining demand will be in the last delivery (second replenishment).

This is determined for every 200 different products to complete the data for product arrival. Now it is known for every day which product and how much of that product will be delivered in the warehouse.

### 5.1.3 Order Lists

The demand per day per product is known for the whole warehouse, but this demand is actually the demand of all stores together. Every store has an order list, these order lists are used by the picker to pick the products. The more stores there are, the more order lists have to be walked by the pickers. This increases the total distance. For comparison of the storage policies it does not matter how many order lists there are, as long as the same order lists are used for every storage policy. In this dataset are 10 order lists created every day, representing 10 different stores.

The order lists are made for the whole year including all 200 products, for every store. Starting with the first store at the first day iterating through every product, it is checked if there is a demand in the warehouse. With a chance of 80 percent a product with demand will be on the order list for that store. The amount of the product that will be picked for that store is normally distributed with an average of the total demand at that day divided by 10 (the number of stores) and a standard deviation of 1 percent of the average. If this number is negative, it will be corrected to zero products. This continues for every product for every day. Now that the demand of the first store is determined, it is subtracted from the demand of the whole warehouse. The procedure is repeated again for the next store but now with a different average for the normal distribution. This average is now the remaining demand of the whole warehouse divided by 9 stores. This continues till the order list for the last store. The remaining demand of the warehouse will be picked for the last store.

This could produce order lists that are not equally divided amongst each other. Especially the last order list will probably have more products to pick compared to the other order lists, but this does not matter. As stated before, the order lists are only used to be able to evaluate the performance of the different storage assignment policies. Besides, it is realistic that the order lists differ in amount of product. Some stores have a bigger capacity and a higher demand than other stores.

### 5.1.4 Product Information

The last data that is necessary is the width per product. The width of a product determines how much space it needs to be stored in the warehouse. There are two different sizes that a product can have, namely 1.5 centimetres for small hangers or 4 centimetres for thick hangers. Small hangers are more common. So with a chance of 75 percent a product gets a width of 1.5 centimetres, otherwise the product will be on a thick hanger. This is determined for every product and does not change.

### 5.1.5 Parameters

There are still some parameters left that should be declared for this dataset. The size of the replenishment warehouse and the parameters for the genetic algorithm of the DSA.

#### The Replenishment Warehouse

The replenishment warehouse will be fixed throughout all datasets and storage assignment policies. The warehouse in this thesis is a three cross-aisle warehouse as is described in Section 1. There are two sides and each rack has two levels. There are 30 racks on each side of the warehouse. Each rack has a length of 20 meters. The aisles between the racks are 25 meters and the middle aisle has a width of 10 meters. It is not possible to store any products on the first 2.5 meters and the last 2.5 meters of an aisle. There are 5 meters between every aisle. The depot is located at the front of the warehouse in the middle between both sides.

#### The Dynamic Storage Assignment

The genetic algorithm in the DSA uses a population size of 100 chromosomes. The mutation rate is 0.05 and the *ReCap* value for the reshuffling is 50 products. For the length of a period are five different values used, 7, 21, 42, 63 and 82 days. Each dataset consist of five instance, one for each length of period. Furthermore, the genetic algorithm continues till the best chromosome has not changed for 10 generations.

## 5.2 Dataset 2

The second dataset consist solely of all year round products, this means that the products are 52 weeks in the warehouse. There is also a demand for these products almost everyday, but it can fluctuate with the time of the year. For instance, some products have a higher demand during the warmer months and a lower demand when it is colder outside. These kind of products are simulated with this dataset.

### 5.2.1 Demand

The demand in this dataset starts at the second day as well. The products arrive for the first time on the first day, but they are stored at the end of the day. So the products can not be picked earlier than the second day. There are 100 products in this dataset. The demand of these products fluctuates between low, mid and high demand. Every 6 weeks the demand of a product can switch between these three demands. The normal distribution with different averages and standard deviations are used to generate these demands. The low demand has an average of 5 products per day with standard deviation of 1. The mid and high demand have an average of respectively 10 and 20, and a standard deviation of respectively 2 and 4. All generated numbers are rounded to its nearest integer. Any negative demand that is generated is corrected to 0. A product has a low, mid or high demand with the chance of 25, 50 and 25 percent respectively. This demand distribution of the products is represented by Figure 6.

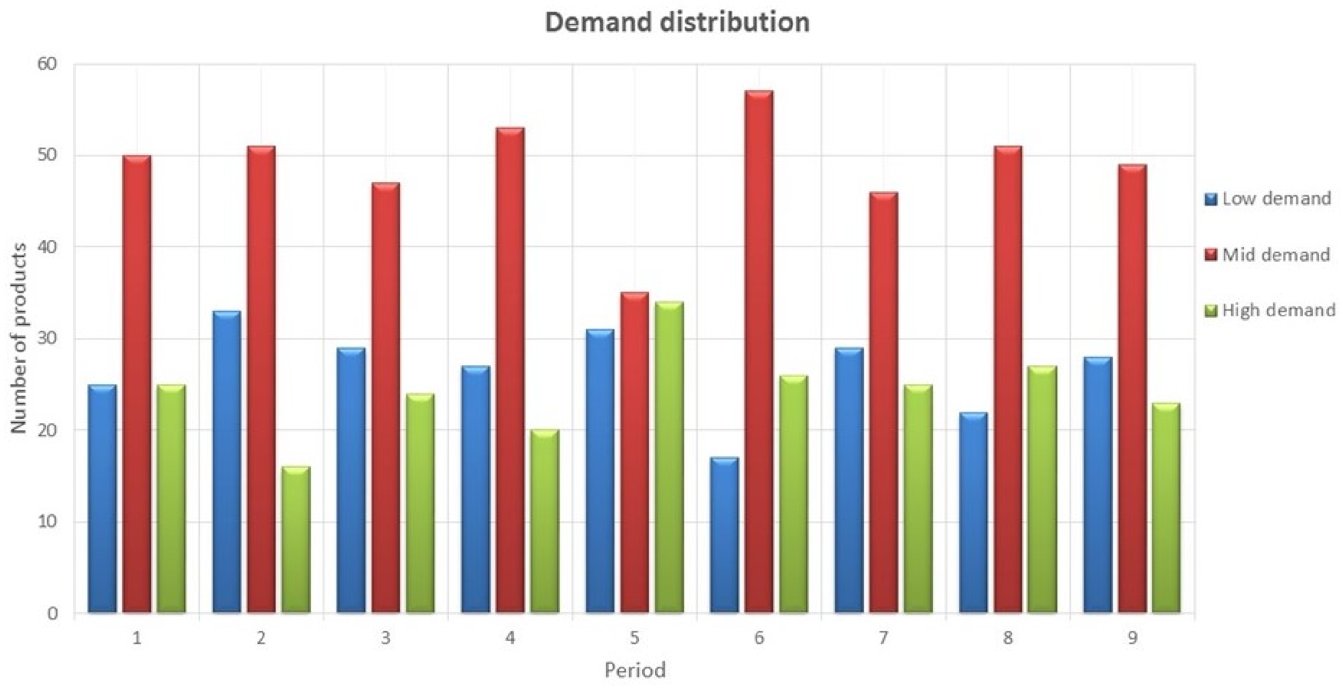


Figure 6: The demand distribution for every period of dataset 2.

From Figure 6 it is shown that the demand of the products fluctuates. Figure 7 gives a better idea of how it fluctuates. For making a clear representation, the demand of only two products are shown.

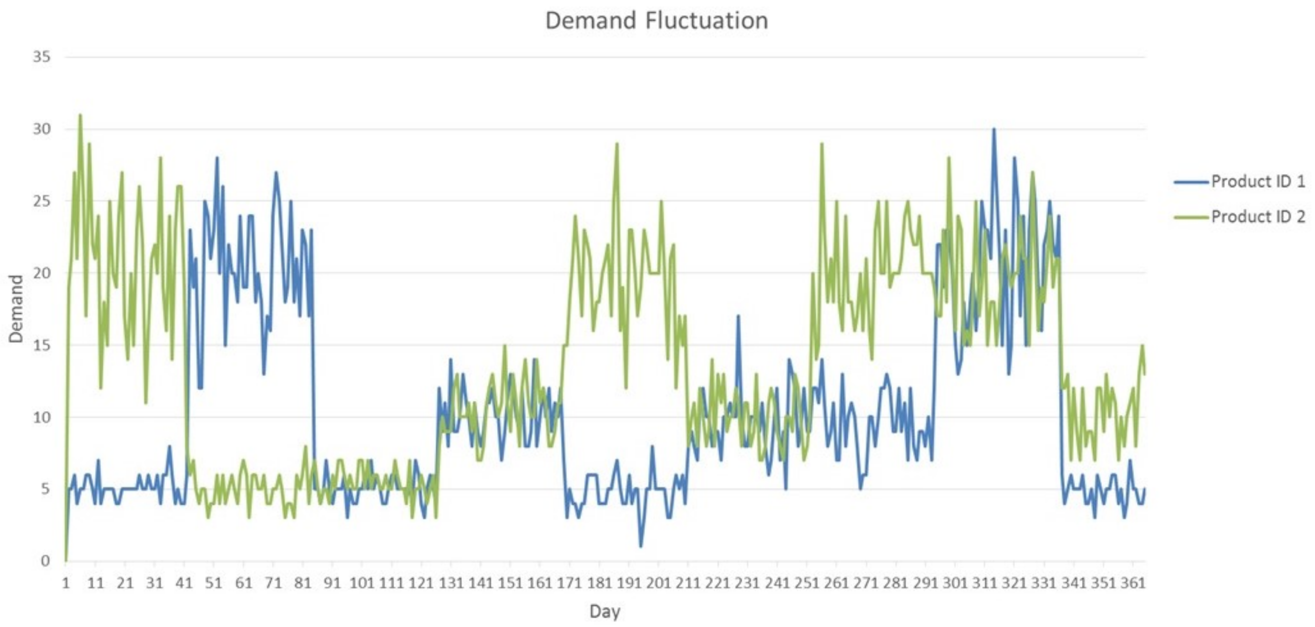


Figure 7: The fluctuation of the demand during the whole year of the first two products of dataset 2.

### 5.2.2 Product Arrival

This dataset has a higher demand than dataset 1. Therefore, there are more products that arrive in the warehouse and products arrive more often. On average a product is 20 times replenished, with a minimum of 13 times and a maximum of 31 replenishments for a certain product. The replenishment days are determined by calculating the cumulative demand of every product. Starting with the demand of the second day, the following days are added until the cumulative demand gets higher than 200 products (demand on the first day is equal to zero, see Section 5.2.1). The day that exceeds a cumulative demand of 200 products is the first replenishment day. Then the cumulative demand is set to zero and the process starts again with the next day to find the following replenishment day. This continues until day 365 is reached. In the same way are the replenishment days for all other products determined. Equation 5.1 gives a vector of the $n$ replenishment days of product $p$ with $RD_0 = 1$, the day that the first products arrive. These days divide the timespan of product $p$ in $n+1$ periods.

$$ReplenishmentDays_p = [RD_0, RD_1, RD_2, \ldots, RD_n] \qquad , \text{for } p = 1, \ldots, 100 \tag{5.1}$$

The day that a product gets a replenishment is now known, but the amount of products that arrive at that day is yet to be determined. The first batch of arrived products will include all the demand upto and including the first replenishment day plus the demand of a random part of the following period. This random part is an integer number of days from zero till one-third of the next period, rounded down. The amount of the first replenishment of products is equal to the demand in the remainder of the following period plus a random part of the next period. This continues till the end of the timespan of 365 days is reached. Equation 5.2 gives a more detailed overview of how the periods are divided that are used to calculate the amount of products that will arrive at the warehouse for the $i$-th replenishment day, with $i = 0, \dots, n$. This is done for every product $p$.

$$PeriodOfDemand_i = \begin{cases} [RD_0, (RD_1 + Random)] & \text{, if} \quad i = 0 \\ [\text{end of } PeriodOfDemand_{n-1}, 365] & \text{, if} \quad i = n \\ [\text{end of } PeriodOfDemand_{i-1}, (RD_{i+1} + Random)] & \text{, otherwise} \end{cases}$$
$$\text{, with} \quad Random = \begin{cases} \text{random number between } \left[0, \left\lfloor \frac{365 - RD_{i+1}}{3} \right\rfloor\right] & \text{, if} \quad i = n-1 \\ \text{random number between } \left[0, \left\lfloor \frac{RD_{i+2} - RD_{i+1}}{3} \right\rfloor\right] & \text{, otherwise} \end{cases} \tag{5.2}$$

## 5.3 Dataset 3

This dataset is a mixture of the first two datasets. The third dataset consists of 100 products of dataset 1 and 50 products of dataset 2.

First a product is randomly chosen from dataset 1. The demand of that product, for every day during the whole timespan, is copied to dataset 3. This creates the demand of the first product in dataset 3. The corresponding data of product arrival and product information from dataset 1 are also copied to dataset 3. So the replenishment days and the amount of products that will be replenished are now exactly the same as the randomly chosen product of dataset 1. This is then 99 times repeated without choosing the same product twice or more. The same process is done 50 times for dataset 2. Then the order lists are created as described in Section 5.1.3 to complete dataset 3.

# 6 Results

In this section are the results described for the three storage assignment policies. The biggest part of the results refer to the Dynamic Storage Assignment, the results of the other two policies are obtained for comparison with the DSA. For the results three different datasets are used and the parameters that are described in Section 5. There are multiple parameters in the program that are interesting to be analysed. But due to the intensiveness of calculation of the DSA, not all parameters will be analysed. The most interesting parameter is the length of a period. This determines how often the DSA has the possibility to reshuffle products.

The results are obtained by writing a Java program using Eclipse. The programming and all the calculation were done on a laptop with an Intel Core i5-4200M processor with 4 GB of RAM. The program ran multiple times for every dataset. Every dataset had five instances, each instance was exactly the same except for the length of a period. Each instance was five times executed, so in total the program ran 75 times with a duration of 80.9 hours. This was only for the first produced dataset of all three datasets. The second and third produced datasets are only used for the SSA and the SAD, because of the long execution time of the DSA. Note that no randomisation was involved with the Systematic Storage Assignment and the Storage Assignment by Demand. So every run of the same instance gave the same results, only the execution times differed. The results for every dataset are presented in separate sections.

## 6.1 Dataset 1

This dataset consist of the seasonal products as described in Section 5.1. First the average results of the Dynamic Storage Assignment are given in Table 2, followed by a discussion of the numbers in this table. Only the first dataset 1 of the three was used for the DSA, because of the long execution time. Subsequently the results of the Systematic Storage Assignment and the Storage Assignment by demand are given and discussed. For these two policies were all three produced datasets 1 used for more reliable results.

**Dynamic Storage Assignment**

For the first instance was a length of 7 days per period chosen. That means that the DSA had

53 periods, 52 periods of 7 days and 1 period of 1 day. The results are given in the first column of Table 2. The DSA gave the same results for every run with different execution times. It was never better to relocate products with this short period. The first chromosome that was produced was also the best one, even for the next generations. That is why there were no more than 11 generations produced.

**Dynamic Storage Assignment**

| **Instance** | **I** | **II** | **III** | **IV** | **V** |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Generations | 11.00 | 14.20 | 178.80 | 152.20 | 140.00 |
| Number of products reshuffled | 0.00 | 463.20 | 233.80 | 148.60 | 124.40 |
| Reshuffle distance (in km) | 0.00 | 67.98 | 33.90 | 16.38 | 17.59 |
| Storage distance (in km) | 5.87 | 6.46 | 6.42 | 6.47 | 6.49 |
| Picking distance (in km) | 809.99 | 635.08 | 567.03 | 582.47 | 593.63 |
| Total distance (in km) | 815.86 | 709.53 | 607.25 | 610.63 | 617.71 |
| Execution time (in seconds) | 137.89 | 129.88 | 1394.97 | 1203.92 | 1128.58 |

Table 2: The average results of five runs of the Dynamic Storage Assignment for every instance with dataset 1.

The second instance made 18 periods for the DSA, 17 periods of 21 days and 1 period of 8 days. For this instance it was better to reshuffle products. Even though this gave extra distance for reshuffling, it reduced the picking distance enough for a lower total distance. This was also the case for the remaining instances. It was for all other instances also more efficient to relocate some products during the year.

The most products were relocated with instance II, but there were also more moments where reshuffling was possible. Instance III had 9 periods so there were only 8 moments to reshuffle. For instances IV and V this were respectively 5 and 4 moments. That is also why there were less products reshuffled there.

The execution time of the instances III, IV and V were longer than instances I and II. Calculating the fitness of every chromosome took up most of the time. For every chromosome in a generation was the whole year simulated. For every product that had to be stored, every product that had to be picked and every product that was reshuffled was the distance calculated that was needed to store, pick or relocate the product. This was done for every day, for each period during the whole timespan of a year. For each new generation this had to be done 50 times for each new offspring chromosome. From Table 2 it is shown that the execution time is positively correlated with the number of generations. The more generation were produced, the higher the execution time was.

Since Table 2 are average numbers, it does not clarify how stable the results were. Therefore, Figure 8 is plotted. This Figure shows the fluctuation of the total distance in kilometres for every instance.

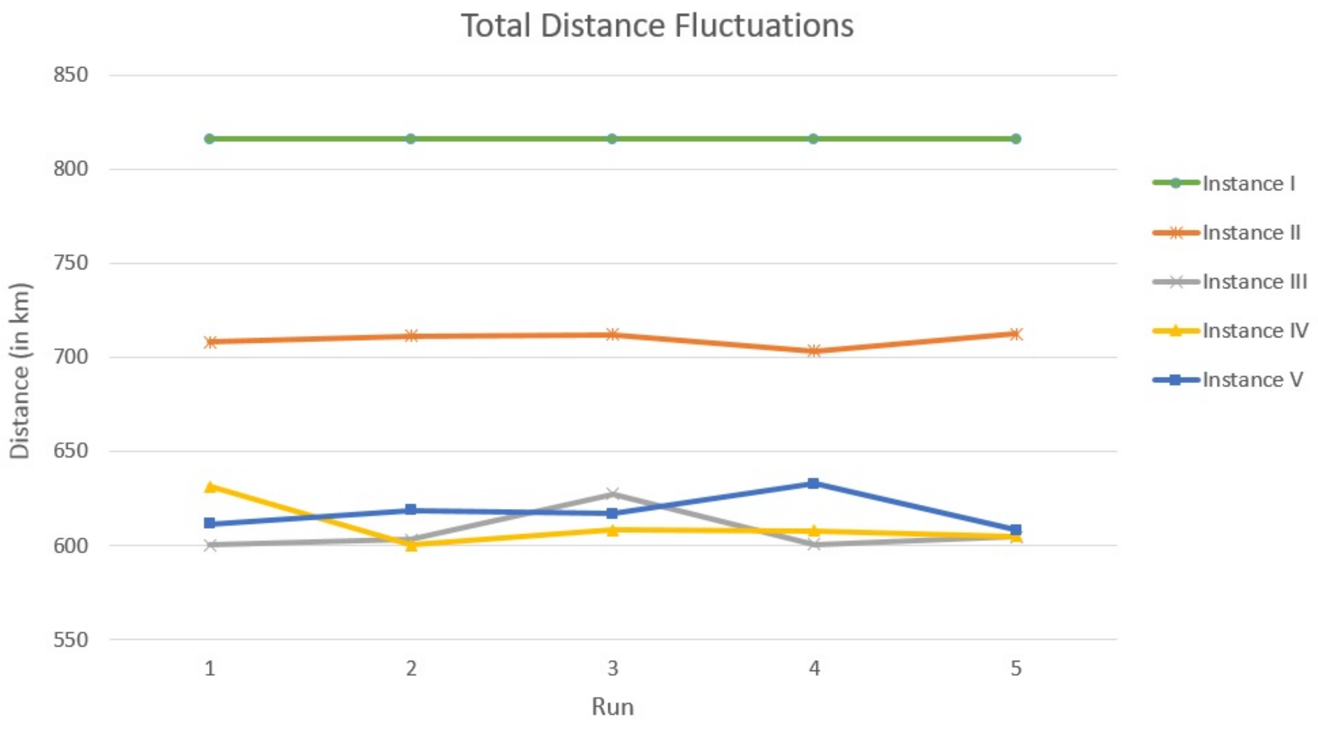


Figure 8: The total distance of every run for every instance of dataset 1 with the Dynamic Storage Assignment policy.

As stated before, instance I gave the same result for every run. The results of the other instances did not differ much from each other. The first run of instance IV is the result that deviated the most from its average, but that was only with 3.4 percent.

**Systematic Storage Assignment & Storage Assignment by Demand**

As is described in Section 5, all datasets are produced three times. Because of the long calculation time of the DSA, only one dataset 1 was used. The other two storage assignment policies are much faster. That is why it was possible to use every three produced datasets 1 for these policies.

These two storage assignment policies did not differ for every run, since there was no randomisation involved for these policies. Only the execution times were different, but they hardly deviated from each other. Storage assignment by demand gave different results for every instance (see Sec-

**Storage Assignment by Demand**

| **Instance** | **I** | **II** | **III** | **IV** | **V** |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Storage distance (in km) | 5.43 | 5.38 | 5.43 | 5.39 | 5.39 |
| Picking distance (in km) | 792.39 | 788.83 | 800.51 | 786.31 | 786.31 |
| Total distance (in km) | 797.82 | 794.22 | 805.94 | 791.70 | 791.70 |
| Execution time (in seconds) | 0.72 | 0.65 | 0.65 | 0.65 | 0.67 |

Table 3: The average results of the three datasets 1 of the Storage Assignment by Demand for every instance that ran five times.

tion 2. That was because the horizon for calculating the demand of a product changed along with the length of the period, i.e., the length of a period was used for calculating the future demand of a product. Note that instance IV and V gave the same results. This was because of the large length of a period and the duration of one seasonal product in the warehouse. One product stayed on average 42 days in the warehouse. There were no products that stayed longer than 63 days in the warehouse, so the demand calculated for each product was the same for instance IV and instance V. That means that the same decisions were made for assigning a storage location to a product.

The Systematic Storage Assignment gave a longer total distance travelled on average than the SAD. Since there was no randomisation involved and nothing changed for every instance, the SSA gave the same result per dataset. The average results of the three datasets 1 are given in Table 4.

| **Systematic Storage Assignment** | |
|---|---|
| Storage distance (in km) | 9.60 |
| Picking distance (in km) | 806.57 |
| Total distance (in km) | 816.17 |
| Execution time (in seconds) | 0.67 |

Table 4: The average results of the three datasets 1 of the Systematic Storage Assignment.

### Comparison of the storage assignment policies

For comparison of the different storage policies for dataset 1 only the first dataset is used. Since DSA only has results for that dataset, it can only compare the other two storage assignment policies with that dataset.

Figure 9 gives the average results for this dataset. From this figure it follows that the best chromosome of the DSA for instance I gave exactly the same results as the SAD. Because the first chromosome was made by following the same rules as the SAD policy. It only took longer for the

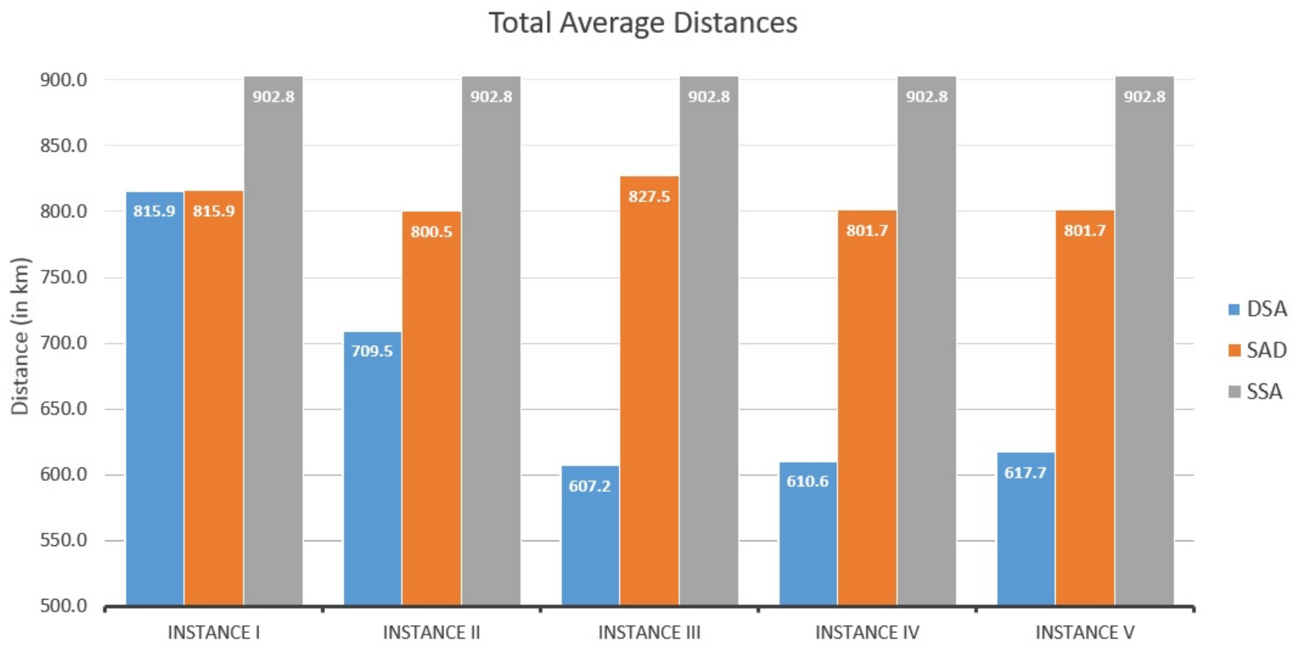


Figure 9: The total average distances of every instance of dataset 1 for the three storage assignment policies. Only the first dataset was used.

DSA to get there. Furthermore, the DSA performed the best over all instances. Since a seasonal product was 42 days in the warehouse, the DSA gave the best results with a period length of 42 days. It even reduced the total average distance with almost 33 percent, compared to the current SSA policy. It did not gained more efficiency when it was possible to relocate a product more than once. That is also because the demand of a seasonal product did not change.

## 6.2 Dataset 2

In this dataset are only all-year round products (see Section 5.2). The results of the DSA are first given, followed by the results of the other two storage assignment policies. Then the three policies are also compared with each other with only the first produced dataset of dataset 2.

**Dynamic Storage Assignment**

The results of the DSA with dataset 2 are comparable with the results of dataset 1, as can be seen in Table 5. The first instance never found a better solution than the first chromosome. Reshuffling products never improved the efficiency in the warehouse when the length of a period was 7 days. For all other instances gave reshuffling products a better solution. With instance II were the most products reshuffled as well, for the same reason. The best result was also with instance III for this dataset. For this dataset also applies that more generations produced, means longer execution times.

**Dynamic Storage Assignment**

| Instance | I | II | III | IV | V |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Generations | 11.00 | 108.20 | 141.40 | 106.60 | 117.80 |
| Number of products reshuffled | 0.00 | 1808.20 | 857.20 | 542.80 | 436.80 |
| Reshuffle distance (in km) | 0.00 | 262.50 | 123.09 | 78.71 | 62.67 |
| Storage distance (in km) | 32.46 | 37.53 | 36.82 | 37.05 | 36.92 |
| Picking distance (in km) | 3513.18 | 3176.28 | 3050.52 | 3096.52 | 3093.34 |
| Total distance (in km) | 3545.64 | 3476.31 | 3210.44 | 3212.27 | 3192.93 |
| Execution time (in seconds) | 916.43 | 7272.36 | 9354.04 | 7068.93 | 7708.39 |

Table 5: The average results of five runs of the Dynamic Storage Assignment for every instance with dataset 2.

The results of the different runs were less stable for this dataset than for dataset 1. The fluctuation in total distance for every run are given in Figure 10. Instance I produced the same result every time. Instance II has an outlier of almost 5 percent deviation from its average at the third run. All other results are closer to its average.

**Systematic Storage Assignment & Storage Assignment by Demand**

For these storage assignments were all three datasets 2 used to obtain the results described in Table 6. These results are also comparable with dataset 1, only the distances are much longer for every instance. This is because the products had a demand during the whole year and not only for 42 days on average. That is also the reason why instance IV and V do not have the same results for dataset 2. The demand of the products now differs from each other, so the SAD made other choices.

**Storage Assignment by Demand**

| Instance | I | II | III | IV | V |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Storage distance (in km) | 32.09 | 32.01 | 32.13 | 31.93 | 32.16 |
| Picking distance (in km) | 3469.83 | 3463.26 | 3466.16 | 3445.13 | 3427.21 |
| Total distance (in km) | 3501.92 | 3495.27 | 3498.29 | 3477.06 | 3459.37 |
| Execution time (in seconds) | 1.87 | 1.91 | 1.88 | 1.92 | 1.90 |

Table 6: The average results of the three datasets 2 of the Storage Assignment by Demand for every instance that ran five times.

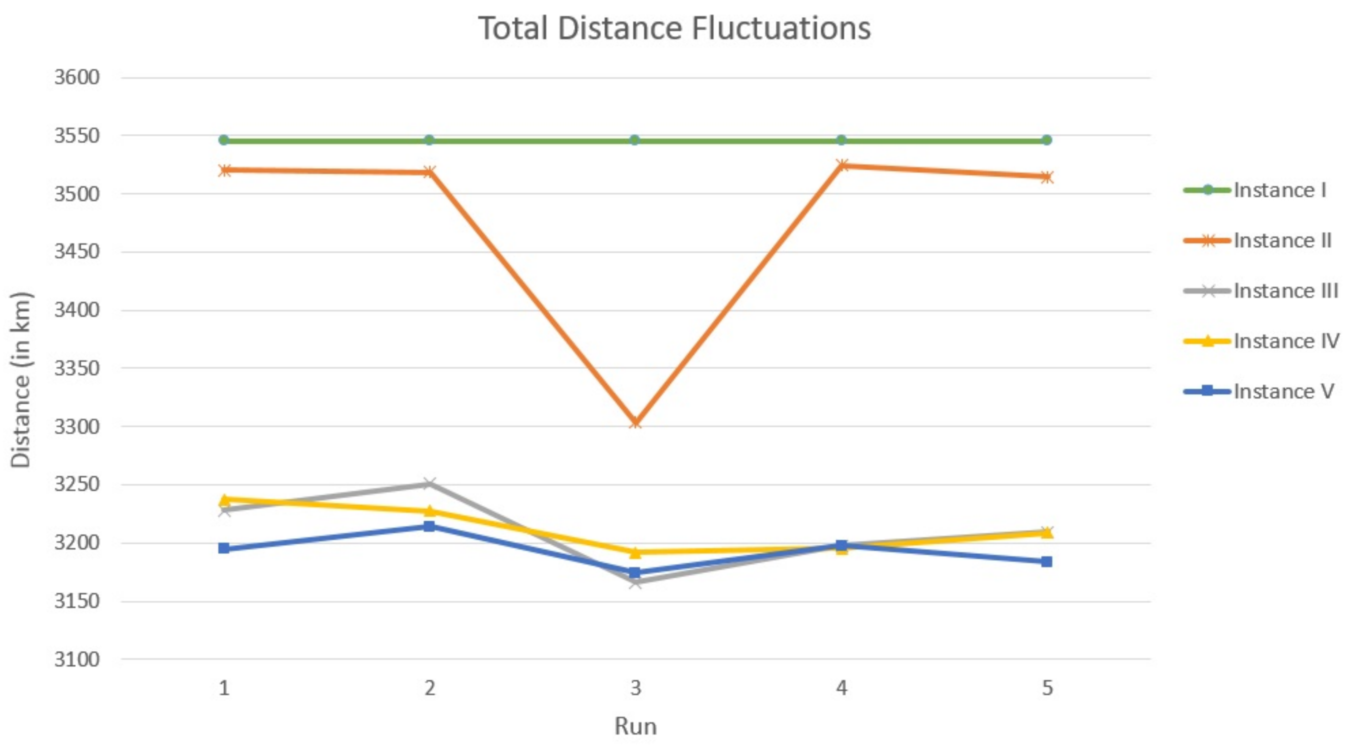


Figure 10: The total distance of every run for every instance of dataset 2 with the Dynamic Storage Assignment policy.

For this dataset, the Systematic Storage Assignment gave shorter distances than the Storage Assignment by Demand. Because of the long demand of the products, more products are in the warehouse at once. Since this dataset is so big that almost the whole warehouse was filled every day, the SAD lost its advantage. All products had to be picked with both policies, so the whole warehouse had to be walked any way. The SSA was now more efficient since the racks closest to the depot are filled up first. The results of the SSA are in Table 7

| **Systematic Storage Assignment** | |
|---|---|
| Storage distance (in km) | 49.47 |
| Picking distance (in km) | 3392.05 |
| Total distance (in km) | 3441.52 |
| Execution time (in seconds) | 1.69 |

Table 7: The average results of the three datasets 2 of the Systematic Storage Assignment.

### Comparison of the storage assignment policies

For this comparison also only the first dataset 2 is used. Figure 11 makes clear that the best result was from the DSA policy for a period length of 84 days. The longest period gave the best result. The DSA seemed to make better choices when it includes the demand of a longer period, although the results of instances III, IV and V are close to each other.

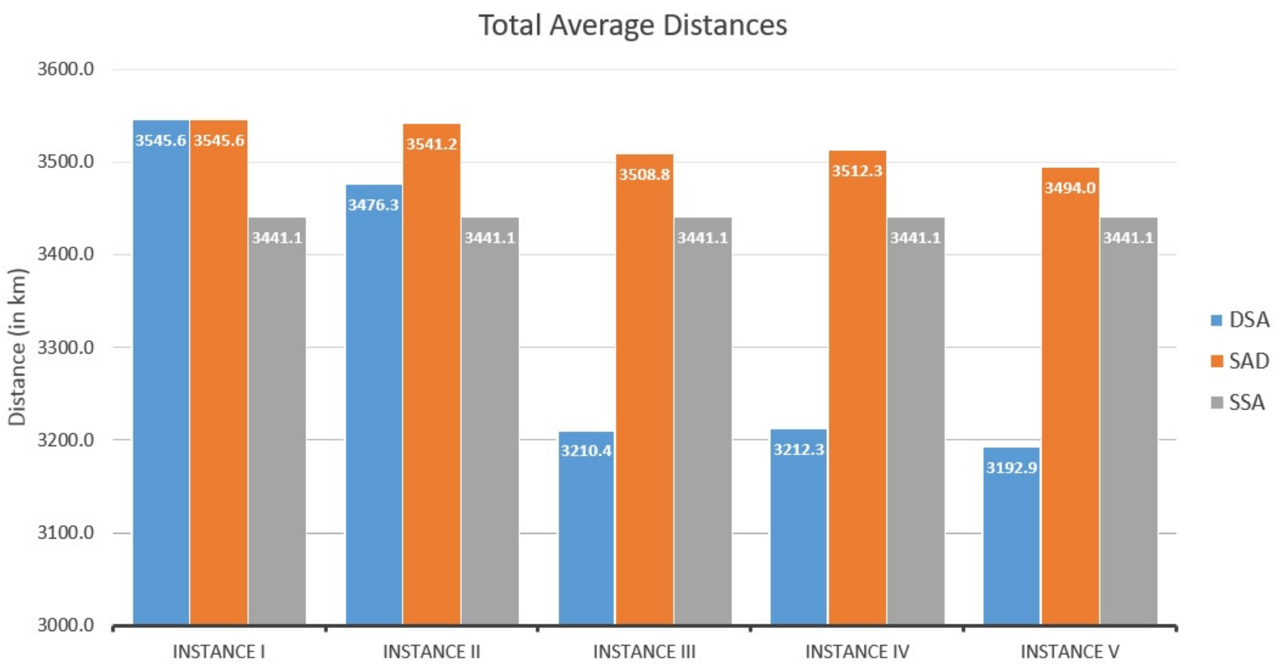


Figure 11: The total average distances of every instance of dataset 2 for the three storage assignment policies. Only the first dataset was used.

For this dataset it is also shown that for all instances it was better to use the SSA rather than SAD.

## 6.3 Dataset 3

This dataset is of a mixture of dataset 1 and 2 (See section 5.3). The average results are first given, then the results of the other two storage assignment policies follow. Finally the three policies are compared with the first produced dataset of dataset 3.

**Dynamic Storage Assignment**

The DSA with dataset 3 gave comparable results with dataset 1 and 2. Only for the first instance was it also never an improvement to reshuffle products. For all other instance the total distance reduced with reshuffling products. The most products were reshuffled with instance II as well and the best result was also obtained with a period length of 42 days. All the average results of the five runs are shown in Table 8.

**Dynamic Storage Assignment**

| **Instance** | **I** | **II** | **III** | **IV** | **V** |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Generations | 11.00 | 233.60 | 294.00 | 103.80 | 122.20 |
| Number of products reshuffled | 0.00 | 1124.80 | 545.40 | 330.20 | 277.40 |
| Reshuffle distance (in km) | 0.00 | 163.13 | 78.57 | 48.38 | 39.99 |
| Storage distance (in km) | 17.66 | 19.92 | 19.80 | 19.73 | 19.81 |
| Picking distance (in km) | 2134.67 | 1752.14 | 1749.72 | 1838.02 | 1844.69 |
| Total distance (in km) | 2152.33 | 1935.18 | 1848.09 | 1906.12 | 1904.50 |
| Execution time (in seconds) | 393.96 | 6682.77 | 8354.10 | 3020.87 | 3488.07 |

Table 8: The average results of five runs of the Dynamic Storage Assignment for every instance with dataset 3.

There are some differences in the total distance between the runs of an instance. The instances II and IV are fluctuating the most. Instance IV has the biggest difference between its minimum and maximum value of almost 100 km. These fluctuations are shown in Figure 12

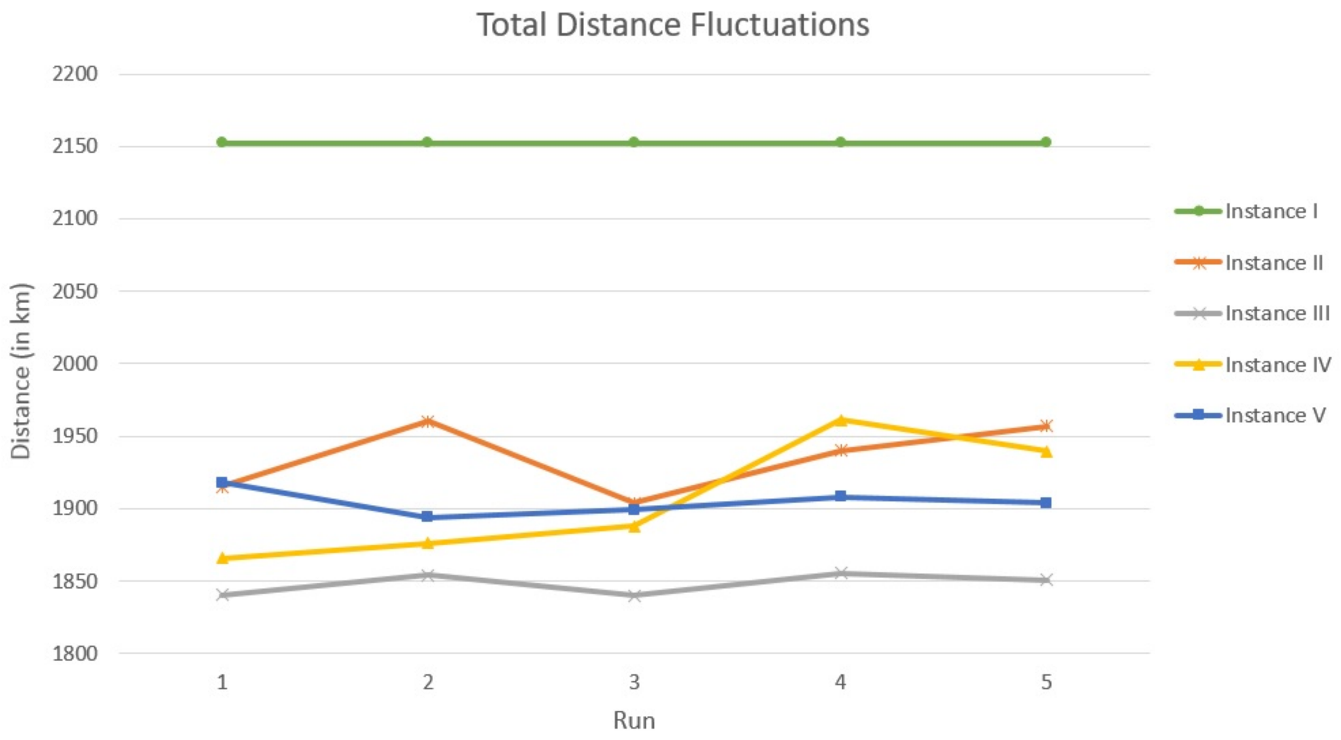


Figure 12: The total distance of every run for every instance of dataset 3 with the Dynamic Storage Assignment policy.

**Systematic Storage Assignment & Storage Assignment by Demand**

To obtain the results for the SSA and the SAD policies, all three produced datasets 3 were used. The average results of the five instances with the SAD policy are shown in Table 9. The results look a lot like the results of dataset 1 and 2. Instance I with the shortest length of period performed worst. The total distances of instances II and III do not differ much. The performance of the SAD policy also improves when the demand of a product is calculated with a longer horizon for this dataset. This is also shown for datasets 1 and 2. But with dataset 3, the SAD performed the best with a period length of 63 days and not with 84 days.

**Storage Assignment by Demand**

| **Instance** | **I** | **II** | **III** | **IV** | **V** |
|---|---|---|---|---|---|
| Length of period (in days) | 7 | 21 | 42 | 63 | 84 |
| Storage distance (in km) | 17.33 | 17.31 | 17.30 | 17.20 | 17.22 |
| Picking distance (in km) | 2125.91 | 2110.66 | 2109.31 | 2053.01 | 2080.31 |
| Total distance (in km) | 2143.24 | 2127.97 | 2126.61 | 2070.22 | 2097.53 |
| Execution time (in seconds) | 1.08 | 1.06 | 1.09 | 1.14 | 1.09 |

Table 9: The average results of the three datasets 3 of the Storage Assignment by Demand for every instance that ran five times.

The Systematic Storage Assignment also performed better than the Storage Assignment by Demand with dataset 3. This is for the same reason. With so many products in the warehouse, it is better to fill up the racks near the depot as much as possible. The average distances of the SSA are given in Table 10.

**Systematic Storage Assignment**

| | |
|---|---|
| Storage distance (in km) | 27.21 |
| Picking distance (in km) | 2064.46 |
| Total distance (in km) | 2091.66 |
| Execution time (in seconds) | 1.02 |

Table 10: The average results of the three datasets 3 of the Systematic Storage Assignment.

**Comparison of the storage assignment policies**

For comparison of the three storage assignment policies only the first produced dataset of dataset 3 was used. All total average distances of the three policies are shown in Figure 13. With the DSA policy the lowest total average distances are reached with every instance. Except for instance I, the SSA has the best results if a period takes 7 days. Furthermore, the SSA is always better than the SAD except with instance IV. The best result was obtained with the DSA for a period length of 42 days.

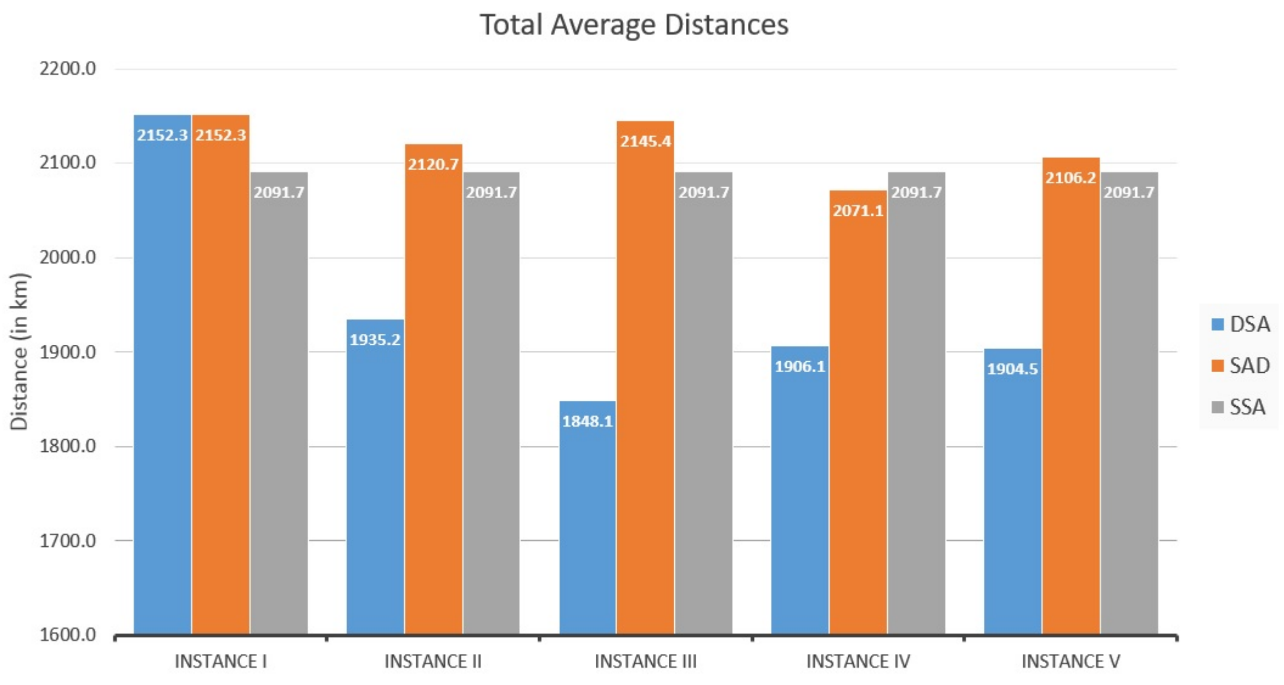


Figure 13: The total average distances of every instance of dataset 3 for the three storage assignment policies. Only the first dataset was used.

# 7 Conclusion

Efficiency in a replenishment warehouse is very important. Wherever something can go more efficient, it should be considered. Especially when it can reduce costs. Labour hours are an important part of the costs in a warehouse. There are many different ways to use labour hours in a warehouse. One way is to use it on the pickers who need to pick all the products from all order lists. The more a picker needs to travel to pick all products from an order list, the more hours it will take to complete all order lists. Reducing the travel distances of the pickers will reduce the labour hours needed and therefore it will reduce the costs of a warehouse. Storage assignment of products can have a big influence on the travel distance of a picker. This travelling distance can be reduced by taking a closer look to the placements of the products in the warehouse.

In this thesis three different storage assignment policies were used to investigate the travelling distances of pickers in a rectangular three-cross-aisle replenishment warehouse. The Systematic Storage Assignment (SSA) policy is currently used in the replenishment warehouse of H&M. The Storage Assignment by Demand (SAD) policy takes the demand of a product into account when assigning the product to a location in the warehouse. The Dynamic Storage Assignment (DSA) policy makes it also possible to relocate products to another location in the warehouse over time. One whole year of 365 days was simulated. During this year new products arrived, products were replenished and products were picked for ten different stores. The three storage assignment policies were tested with three different datasets. Each dataset had five different instances, each instance had a different length of a period. For the first dataset it turned out that the SSA was performing the worst. The DSA managed to lower the total distance for every instance. For dataset 2, the SSA had a lower total average distance for every instance than the SAD. The DSA managed to have a the lowest total average distance in three of the five instances. With the third dataset the DSA was in four of the five instances the best performing storage assignment policy.

From this research follows that the SSA did not perform good when there was a lot of room in the warehouse available, compared to the other two policies. The SAD managed to make better use of the available space in the warehouse. This changed when more products were in the warehouse and less space was available. Then the SSA managed to get lower total average distances than the SAD. If it was possible to relocate products in the warehouse after a period of time to another location in the warehouse, it could reduce the total average travel distance of the pickers. Therefore, the DSA was the best performing storage assignment policy with every dataset, provided that the length of a period was at least 21 days.

The demand of a product can change over time. A product can start with a low demand and have a high demand later in the year, or the other way around. The demand can also fluctuated throughout the whole year. The SSA and the SAD policies are static storage assignment policies. That means that a product get only once a location assigned and it will remain there till the last product is picked. Therefore, these policies do not take the demand fluctuation into account. The DSA looks at the demand of the upcoming period and assigns a location to the product according to that demand period. At the end of every period, a product can get a different location according to their upcoming demand. This possibility makes the DSA a better performing storage assignment policy for these datasets.

# 8 Future Research

In this section are some suggestions given for a follow-up study or improvements of the DSA as described in this thesis.

This thesis introduced a new storage assignment policy, the Dynamic Storage Assignment. The biggest flaw of the DSA is its calculation time. The DSA simulates the whole given timespan with arriving products and products to be picked. Every meter that is necessary to store, to reshuffle and to pick the products is calculated for every day, for each period. The way that this is handled in this thesis, is very time consuming. Reducing the calculation time could make the DSA more useful in practice.

The DSA has a lot of different parameters thanks to the genetic algorithm that is used. These parameters, such as the population size or mutation rate, can be analysed to improve the performance of the DSA. Another way of selecting the parent chromosomes, reproduction and evolution could be used to give better and faster results.

The horizon that was used for calculating the demand of a product was the same as the length of a period in this thesis. This can be analysed by making the upcoming number of days independent of the length of a period. Especially for the instances with a short length of period could this have a lot of influence. Extending the horizon for calculating the demand of a product could lead to smarter decisions that the DSA will make.

Another way to improve the DSA is to handle the different locations of the same product. When a replenishment of a product arrive, the DSA will first try to place the arrived product in the same location as the product already has in the warehouse. If this is not possible, it will get a another location. Better results could be obtained if it is possible to merge these different locations, during the reshuffling. The DSA does not have this possibility yet.

## References

Chan, F. T. and Chan, H. K. (2011). Improving the productivity of order picking of a manual-pick and multi-level rack distribution warehouse through the implementation of class-based storage. *Expert Systems with Applications*, 38(3):2686–2700.

De Koster, R., Le-Duc, T., and Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182(2):481–501.

Hausman, W. H., Schwarz, L. B., and Graves, S. C. (1976). Optimal storage assignment in automatic warehousing systems. *Management science*, 22(6):629–638.

Ho, Y.-C., Su, T.-S., and Shi, Z.-B. (2008). Order-batching methods for an order-picking warehouse with two cross aisles. *Computers & Industrial Engineering*, 55(2):321–347.

Ho, Y.-C. and Tseng, Y.-Y. (2006). A study on order-batching methods of order-picking in a distribution centre with two cross-aisles. *International Journal of Production Research*, 44(17):3391–3417.

Hsu, C.-M., Chen, K.-Y., and Chen, M.-C. (2005). Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Computers in Industry*, 56(2):169–178.

Jarvis, J. M. (1984). Optimal product layout in an order picking warehouse.

Malmborg, C. J. and Bhaskaran, K. (1990). A revised proof of optimality for the cube-per-order index rule for stored item location. *Applied Mathematical Modelling*, 14(2):87–95.

Muppani, V. R. and Adil, G. K. (2008a). A branch and bound algorithm for class based storage location assignment. *European Journal of Operational Research*, 189(2):492–507.

Muppani, V. R. and Adil, G. K. (2008b). Efficient formation of storage classes for warehouse storage location assignment: a simulated annealing approach. *Omega*, 36(4):609–618.

Petersen, C. G. (1997). An evaluation of order picking routeing policies. *International Journal of Operations & Production Management*, 17(11):1098–1111.

Petersen, C. G. (1999). The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations & Production Management*, 19(10):1053–1064.

Petersen, C. G. (2002). Considerations in order picking zone configuration. *International Journal of Operations & Production Management*, 22(7):793–805.

Ratliff, H. D. and Rosenthal, A. S. (1983). Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. *Operations Research*, 31(3):507–521.

Roodbergen, K. J. and De Koster, R. (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 133(1):32–43.

Thonemann, U. W. and Brandeau, M. L. (1998). Note. optimal storage assignment policies for automated storage and retrieval systems with stochastic demands. *Management Science*, 44(1):142–148.

Zhao, Y., Shi, Y., and Karimi, H. R. (2012). Entry-item-quantity-abc analysis-based multitype cigarette fast sorting system. *Mathematical Problems in Engineering*, 2012.