Solving the storage location assignment problem and the order batching problem at an e-fulfillment center

Martens, W.

Award date:
2021

Solving the storage location assignment problem and the order batching problem at an e-fulfillment center

By
W. Martens

In partial fulfillment of the master's degree in
Operations Management and Logistics

First supervisor: Dr. R.A.C.M. Broekmeulen

Second supervisor: Dr. A. Marandi

Third supervisor: Dr. M. Mirzaei

Student identification number: 0905348

Department of Industrial Engineering & Innovation Sciences

Eindhoven University of Technology

05-04-2021

Abstract

Warehouses that act as e-fulfillment centers face many challenges. These warehouses have to handle many customer orders that have to be retrieved from a large product assortment in a short time period. For such a warehouse, it is important to optimize the picking process to minimize the order picking times and, with that, picking costs. The Storage Location Assignment Problem (SLAP) and the Order Batching Problem (OBP) are two problems that need to be solved to improve the picking process. SLAP optimizes the location assignments of the Stock Keeping Units (SKUs) in the warehouse whereas OBP optimizes the picking batches that are made out of customer orders. In this study, these two problems are investigated at an e-commerce warehouse that handles customer orders with relatively large order sizes. To solve the SLAP, SKUs are allocated to warehouse sections using two different methods: the frequency and the volume-based turnover policy. A greedy construction and hill-climbing local search algorithm are then used to seek an improved allocation in terms of distance traveled per order. The OBP is solved by using an algorithm that first selects all orders in the same warehouse section and then uses a similarity coefficient to batch orders from that selection that visit the same locations. In a case study at Kuehne + Nagel, it is shown that allocation can be improved significantly using local search. However, since the demand for SKUs, in this case, is unpredictable and orders can contain many different SKUs, the allocation decisions based on one period do not significantly decrease the traveling distances for the next period. The batching method, however, does decrease picker travel distance significantly.

Executive summary

This study addresses the Storage Location Assignment Problem (SLAP) and the Order Batching Problem (OBP) that is solved in one of the warehousing systems at Kuehne + Nagel in Veghel. In this research, SLAP assigns items to zones in the warehouse such that the most picked items are in the front section of the warehouse. Different methods of solving SLAP are tested in a case study at the company and compared with the method that is currently used to assign products to locations in the warehouse. The OBP groups customer orders into pick batches, such that the total distance required to pick the items of one pick batch is minimized. To minimize the travel distance per order, an algorithm is developed that groups customer orders into pools and uses the current batching method within those order pools. The knowledge obtained from this research gives insights on how to minimize travel distance per order.

Problem Formulation

Given the many and large customer orders that are received, large assortments, unstable demand, large storage areas, and tight delivery schedules, e-fulfillment centers such as Kuehne + Nagel Veghel ask for efficient order picking systems. Reducing the distance that pickers have to travel to pick customer orders helps the company to cope with the above whilst not making any large investments. The performance issues, indicated through data analysis, of the current way of allocating SKUs to storage locations and order batching ask for a new logic such that the current order picking travel distance will be reduced. The goal of the research is to "Improve order picking efficiency by optimizing the storage allocation and batching method". To achieve the goal the research question of this study is defined as follows: *How to decrease order picking travel distance by solving the storage location assignment problem and order batching problem?*

Model and solution approach

The model in this study solves the storage location assignment problem and order batching by minimizing the total distance that is traveled per order. Within the model, several assumptions and constraints are described. Allocation is done after each period of 28 days using the data of that period. This allocation will be used to pick orders in the following period. Customer orders that are received by the warehouse on one day are picked the next day. Therefore, batches are made after receiving all orders for that day. The warehouse layout consists of two aisles with three cross-aisles, of which two are located at the ends of the aisles. Pickers can traverse the aisles using two different routes. One of the routes passes all items in the warehouse and one of them only travels through the front section. Customer orders that are received consist of a maximum of 16 units and batches are made of a maximum of six customer orders. One batch is picked in a single picking route.

Several methods to solve the assignment problem are analyzed in this report. Two assignment policies are used that use demand volume and frequency to allocate products to zones in the warehouse. Next to these policies, a greedy method is developed to attempt to solve the quadratic assignment problem that is defined when considering the chances of SKUs being ordered together. A local search algorithm is also created and employed to solve the SLAP. The order batching problem is solved using a batching method that batches the incoming customer orders based on the zones in which the items in the orders are located. Within those order pools, a similarity coefficient is used to batch orders that are similar to one another. This research is not only focused on finding the optimal method of solving the problems. It also analyses the performance of the output that results from solving the problem by the means of a case study.

Results

The SLAP at Kuehne + Nagel is complex since the demand is very unstable in this case. The policy that allocates items based on the demand frequency does not outperform the currently used policy that allocates items based on the demand volume. The greedy algorithm that uses the chance of item pairs being ordered together did show positive results in several periods but did not significantly outperform both policies. The local search algorithm did significantly improve product allocation for one period based on data of the same period. In practice, however, the allocation is used to find an allocation for the following period. Because of the unstable demand, the local search algorithm showed small improvements that were insignificant.

The algorithm used to create batches does bring significant improvement to the system. This batching method increased the number of batches that only contained items that are picked in the front zone of the warehouse relative to the total number of batches picked by 27%. The total travel distance was decreased by 9.1% because of this. In the analyzed time period in 2020, pickers would have traveled 166,860 meters less in the case that this batching method was used.

Recommendations

Based on the insignificance of the positive effect brought by the allocation methods, it is not recommended to the company to solve the SLAP using either the greedy algorithm or the local search method that is tested in this thesis report. To create a better allocation method, the company could make improvements in communication about SKU availability and the promotions that could cause customers to order certain item pairs together. Because of the results described in this thesis, it is suggested that the company should reconsider its batching method. Using the batching algorithm described in this report decreases the picker travel distance.

Preface

This report is not only the result of my master thesis project, but it also concludes my master's program Operations Management and Logistics. This marks the end of seven unforgettable years as a student at the Eindhoven University of Technology. During these seven years full of new experiences I have met many interesting and inspiring people and made friendships that will last forever. At the end of my study, the Covid virus impacted many student lives due to the restrictions that are currently still imposed. I am grateful that I could live almost my entire student life without these restrictions to enjoy it to the fullest. I would like to use this preface to thank everyone who has supported me in the completion of this master thesis project. The order in which individuals are mentioned is irrelevant since everyone has contributed to support me in his or her own way.

First of all, I would like to thank my supervisor from the TU/e Rob Broekmeulen, for his time, effort, and guidance throughout this process. The knowledge you have in your field of expertise has inspired and helped me throughout this project. I appreciate your insights during our meetings and valuable and critical feedback. You were always kind and correct in your way of communicating with me and a very pleasant person to work with during the last couple of months of my study. Secondly, I would like to thank Ahmadreza Marandi for being my second supervisor. Your enthusiasm, knowledge, and tips inspired and helped me. Thank you for your feedback during my thesis project.

Next to my supervisors at the TU/e, I would like to thank Kuehne + Nagel as a company and more specifically the Planning and Control and Business Support teams in Loods 9 in Veghel. I want to thank Bas van de Burgt, my company supervisor at Kuehne + Nagel. Your knowledge in the field of warehousing and operation management has been of great help during this project. Moreover, your enthusiasm and the inspiring meetings that we had led me to decide on pursuing a career in this sector.

Finally, I would like to express gratitude to my family, friends, and my girlfriend. You all supported me by taking the time to listen to me, keeping me focused, and motivating me during my study and my master thesis project.

To conclude this preface I would like to wish you a lot of fun reading this report.

Willem Martens

Contents

List of Figures

List of Tables

List of abbreviations

FoO – Frequency of Ordering
K+N – Kuehne + Nagel
OBP – Order Batching Problem
QAP – Quadratic Assignment Problem
SKU – Stock Keeping Unit
SLAP – Storage Location Assignment Problem
TOS – Turnover Based Storage

1. Introduction

In this section of the report, the topic of the research will be introduced and explained. First, a short introduction to e-fulfillment is given. Second, some key concepts that are used throughout this thesis are elaborated. These concepts include the storage location assignment problem and the order batching problem. Last, the structure of this report is specified.

1.1 E-fulfillment

The e-commerce industry has been growing for the last few years. Worldwide, sales in the industry have grown from 1.3 trillion in 2014 to 3.5 trillion in 2019. Those sales are projected to grow to 6.54 trillion US dollars in 2023 (Clement, 2019). This growth makes for added pressure in the industry. The increase in sales volume gave rise to e-fulfillment warehouses. These warehouses are tailored to the needs of e-commerce retailers. Most e-fulfillment warehouses directly serve the demand of the end-customer in the business-to-customer segment.

E-fulfillment is essentially the storage, picking, packing, and shipping of online orders. The assortment that is offered by online retailers can be much larger than the assortments in traditional brick-and-mortar retailers. The assortment of online retailers is handled by e-fulfillment centers. These centers do not only cope with the added pressure of the size of the assortment, but also with time pressure. Nowadays, customers expect their orders to be delivered to their homes the next day, or sometimes even the same day. This results in tight delivery schedules and adds pressure to the warehousing activities in these tailored warehouses.

To compete in this growing and complex market, e-fulfillment centers must efficiently handle their warehousing activities. One of the most important warehousing activities is the order picking process. This process accounts for 55% of the total operating costs in these warehouses (Bartholdi & Hackman, 2011). Order picking must be continuously improved in order to compete. This also applies to the warehouses of Kuehne + Nagel. Within the scope of this research, the order picking process in one of the warehouses at Kuehne + Nagel in the Netherlands is analyzed. Methods for improvement of the storage location assignment process and order batching process at this company will be suggested and tested to analyze whether these methods decrease picker travel time and therefore effectively improve the order picking process.

1.2 Research topic

As mentioned above, this research firstly focuses on the storage location assignment. The Storage Location Assignment Problem (SLAP) is thoroughly discussed in literature. SLAP concerns the allocation of Stock Keeping Units (SKUs) into storage space in a warehouse system and optimizing the material handling costs or space utilization (Reyes et al., 2019). Many parameters such as the warehouse layout, the available storage space and capacity, item characteristics, and demand behavior affect the SLAP. Due to all these parameters, many variations of the SLAP exist. The SLAP can be solved in many different ways. Methods used to solve the SLAP are: exact methods, heuristics, meta-heuristics, simulations, policies and rules, multi-criteria methods, and other trends and support tools. This research investigates the SLAP at an e-commerce warehouse.

The second topic which this research focuses on is the Order Batching Problem (OBP). The OBP usually consists of two subproblems: the batching of the orders, and the subsequent routing of the resulting batches. The first subproblem deals with deciding on combining sets of customer orders into batches. The second subproblem deals with defining the route to collect the orders in that batch. The OBP is dependent on parameters such as batch capacity and warehouse layout. Object functions usually look for the configuration of batches that minimize either the time, distance, or cost to collect all orders. In

this research, a simple variant of the OBP will be solved since the considered warehouse system deals with routing constraints, making the second subproblem trivial.

1.3 Report structure

In this thesis, it is investigated how the picking process can be improved through improvements in the SLAP and the OBP. A local search algorithm and a greedy algorithm are developed and tested. These can be applied to the situation at Kuehne + Nagel and might be applicable in other situations. In Section 2 of the report, the project environment is given. Section 3 states the problem, research objective, and research questions. Literature research is conducted in Section 4 to gain knowledge and formulate the conceptual design in Section 5. In Section 6, a detailed description of the methods used to solve the problem is stated. Section 7 describes the results in the setting of the case study. Conclusions, limitations, and suggestions for further research are given in Sections 8 and 9 of the report.

2. Project environment

2.1 Kuehne + Nagel: Veghel

Kuehne + Nagel is one of the world's leading logistics providers. The company's main activities are service logistics for organizations which will be referred to in this report as "customers" of K+N. One of the locations at which K+N is active in the Netherlands is Veghel. Contract logistics, overland, warehousing, co-packing, customs, pallet management, and international transport management are services delivered by Kuehne + Nagel Veghel. The establishment specializes in factory operations, business to business (B2B) domestic, B2B International travel retail (ITR), and business to consumer (B2C); referred to as e-commerce in this report.[1] Warehouse 9 handles contract logistics for several large customers. The two different classes of warehousing that are found in this warehouse are B2B ITR and B2C e-commerce. Every type of customer has specific demands, which makes the combination of the two different classes of warehousing difficult. The research conducted in this project focuses on the processes at the above-mentioned warehouse 9 of Kuehne + Nagel Logistics B.V. in Veghel. This warehouse can be divided into different sections that handle different products. This project focuses on the storage and order picking systems and strategies of the section of the warehouse that handles the storage, picking, and packing of packs of beers that are customized and ordered by customers.

2.2 Current situation

2.2.1 Order picking process

In the current system, order pickers travel through aisles to collect and retrieve items that are requested from storage locations within these aisles to fulfill customer orders. The system is defined by a high number of different SKUs (Stock Keeping units) (>1000). Empty locations are refilled with case packs. Pickers are able to pick all items from the storage racks within the pick aisles without the use of cranes or other tools. The current situation is therefore described as a low-level picker-to-parts warehouse in which a batch of six customer orders is collected in a single picking route (Dallari et al., 2009).

A customer order is a set of items that is requested by a single customer. In the situation at Kuehne + Nagel, one customer order consists of a maximum of 16 units. Units are defined as the quantity of items that are picked (i.e. two units of one single SKU can be picked). In this thesis, six is the maximum number of customer orders that can be carried on a product carrier by order pickers. Six boxes are put on the carrier for the picker to sort and start packing the orders while picking. Figure 1 illustrates the carrier that is used while picking. The carriers get too heavy if more orders are picked according to health and safety regulations. It is assumed that these carts are also used for this research. Therefore, order batches are made with batches of a maximum of six orders.

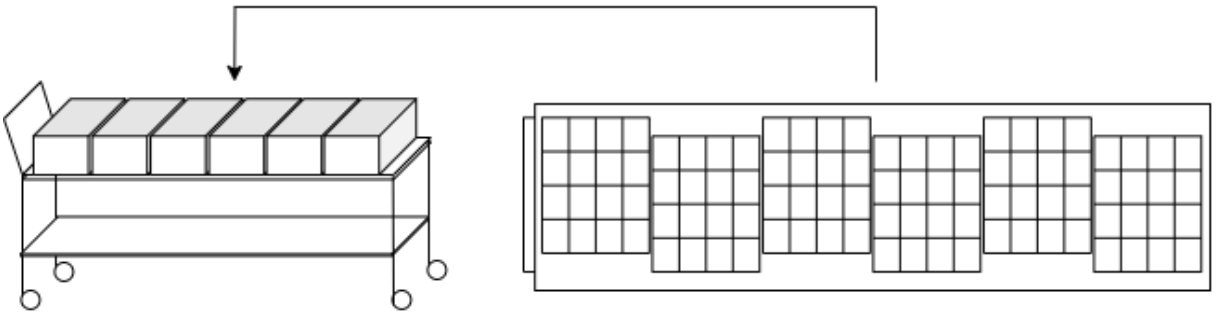

Figure 1: Carriers used for order picking with the top-down view

[1] https://nl.kuehne-nagel.com/nl_nl/top-links/onze-locaties-in-nederland/

Layout

The warehouse section that is in the scope of this thesis contains two pick aisles. The aisles are currently divided into seven zones: P10, P20, P30 … P70. Zone P60 and P70 have larger locations for the fast-moving SKUs and zones P10 and P20 are for returned SKUs without barcodes. The rest of the zones have the same location size. Two case packs of an SKU can be placed in the locations. The SKUs are assigned to these zones based on their sales volume. The layout is visible in Figure 2. A cross-aisle is put just past the middle of each aisle to prevent pickers from walking through the entire aisles when this is not needed. At the beginning and the end of the aisles, there are also cross-aisles, as indicated in the figure. In this research, the part of the aisles between the depot and the middle ross aisle is referred to as zone A (or the front section). The part of the aisles starting from the middle cross-aisle to the end of the aisles is referred to as zone B (or the back section). Pickers always start their route at the depot, which is located at the start of both picking aisles and indicated by I/O in Figure 2.

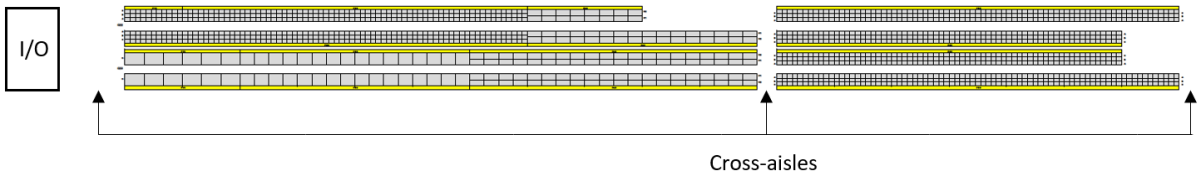

Figure 2: Layout of pick aisles

Routing

At the depot, pickers take an empty product carrier and travel through the pick aisles to collect the orders on the carriers and return to the depot. They use scanners that show the next location to which the picker has to travel. At each of the locations, the scanner shows the quantity of the SKU that has to be picked at that location for each order in the batch. Figure 3 shows the picking route that is traveled. The blue arrows indicate the shortest route that is currently traveled. The red arrows show the extra route that has to be traveled when an SKU behind the middle cross-aisle has to be picked. Due to small aisles and routing constraints, one of these two routes has to be traveled.

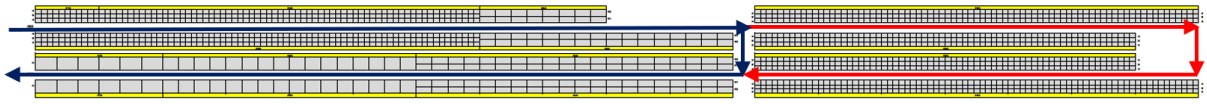

Figure 3: Picking route

2.2.2 Allocation decisions

SKUs are now allocated to zones using a turnover-based policy. This policy assigns SKUs to zones according to the sales volume of that SKU in a certain time period. Currently, the allocation analysis is done manually on a monthly basis. SKUs are assigned to new zones at the beginning of a new month based on the sales volume of the previous month. Only when a location of an SKU is empty, it will be refilled in its newly assigned zone. This is known as passive reallocation. In practice, around 400 locations are available in the front section of the warehouse for the fastest-moving SKUs. In this research, the number 400 is therefore used as the number of available locations in the front section of the warehouse. The literature section in this thesis will further explain the turnover-based storage policy along with other storage assignment policies.

2.2.3 Order batch generation

When all orders for a specific day are received, order batches are created for pickers to collect. Orders are batched using an algorithm that tries to minimize the number of visits to unique locations for

pickers. This is done by calculating a coefficient that indicates the similarity of customer orders. The content of an order can be seen as a collection in which the elements are the SKUs in the order. The content of 2 orders X and Y can be shown in a Venn-diagram:

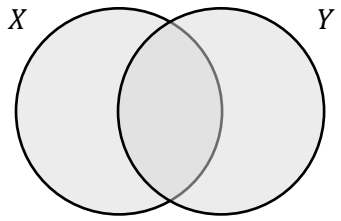

Figure 4: Venn-diagram of order collection and union of orders X and Y

The complete grey area represents all elements from collection X and Y together. This is the union of the two sets and is noted as $X \cup Y$. The overlap of the circles forms the subset of elements that the two sets have in common. This section is the cross-section and is noted as $X \cap Y$. The more SKUs the orders have in common, the more the circles overlap. To create the order batches it is therefore desirable that the cross-section, relative to the union, is as large as possible. For orders X and Y the Jaccard coefficient is calculated using (1):

$$Jaccard\ coefficient = \frac{|X \cap Y|}{|X \cup Y|} \quad (1)$$

Orders with a high similarity coefficient are batched. This will lead to orders with high overlap in picking locations at the beginning of the batching procedure but will result in batches that are totally different from each other towards the end of the procedure.

The current batching algorithm uses the following three steps to create the picking batches from a given set of customer orders:

Step 1: Create a starting pair
Calculate the Jaccard coefficient for each pair of orders left in the given set of customer orders and use the combination of orders with the highest coefficient as the starting pair or seed for the current pick batch.

Step 2: Add orders to starting pair
Calculate the coefficients of each order left in the order set with this starting pair and add the order with the highest coefficient to the current pick batch. If there are multiple orders with the same coefficient, the order with the lowest order ID will be added to the current pick batch.

Step 3: Check if the batch is completed
If less than six orders are in the current pick batch and we still have unassigned orders in the set, go to step 2. If a total of six orders are in the current pick batch, close and release the current pick batch, remove the assigned orders from the order set and return to step one if there are still orders in the set. If not, stop the algorithm.

2.3 Data analysis

Data analysis has been conducted using data from January 2020 until November 2020. In this section, the results of this analysis are shown. The data analysis gives a general idea of the descriptive statistics of SKUs and orders. The analysis indicates some issues with the current system that are discussed in the next section of the report.

Demand statistics

The number of SKUs in the assortment decreased from 1149 to 524 in 2020 and no information about when SKUs are removed from the assortment is given, which could be problematic for the allocation process. The number of orders however did not decrease over the year. Figure 5 shows the number of orders per period (of 28 days) in 2020. The figure illustrates that the number of orders fluctuated substantially over the year. Not only the orders but the demand per SKU also substantially fluctuates. In Figure 6, examples of the demand frequency per period of 5 randomly selected SKUs are shown. Marketing and promotions could cause fluctuations in demand for some SKUs. In this case, however, most SKUs have a demand pattern that is equal to the examples. This implicates unpredictability of demand.

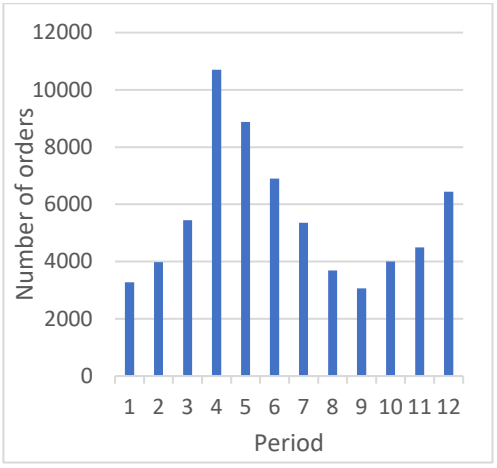

Figure 5: Number of orders per period

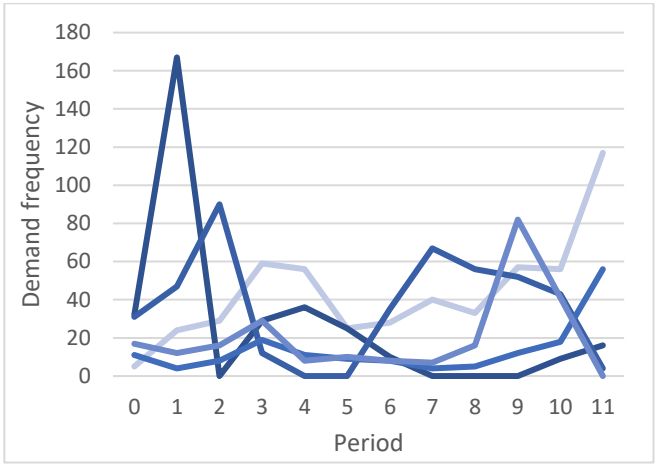

Figure 6: Demand frequency per period of 5 random SKUs

The so-called "long tail" is analyzed to see what small set of SKUs are responsible for a large portion of the demand frequency. Figure 7 shows the division of demand frequency for all SKUs in the assortment over the year 2020. The figure shows that the decrease in frequency flattens around the top 200. Further analysis shows that the top 200 SKUs are responsible for 40.5% of the total sales. The top 200 SKUs, therefore, have a large chance to be allocated in the front section of the warehouse in which the top 400 SKUs are allocated monthly. According to the data, 183 SKUs were allocated in the front section of the warehouse at least eight of the twelve periods that were analyzed.

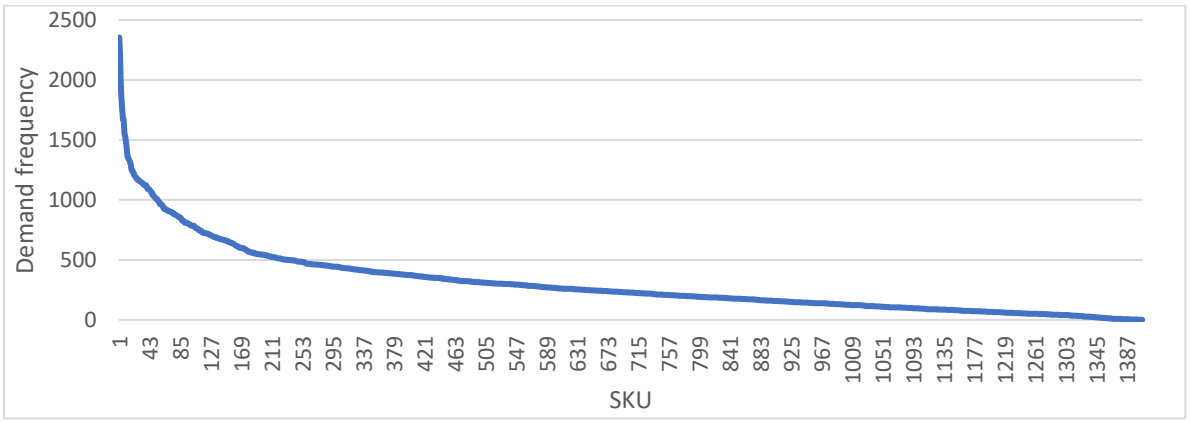

Figure 7: Demand frequency per SKU sorted from most frequently ordered to least frequently ordered SKU

Order statistics

The mean order size in this warehouse is high, but it decreased from 15.4 units in the first month of 2020 to 9.4 in November. The order size is significantly higher than in typical e-commerce warehouses (1-3 units per order). Currently, the allocation method uses the sales volume of each SKU to allocate the SKUs in the warehouse. This method has proven to be efficient in practice in comparable warehouses with low order sizes. In this warehouse, multiple units of a single SKU are often ordered. Figure 8 shows the number of different SKUs that orders contain. The mean number of SKUs per order is much higher than in comparable warehouses, namely 6.5. Because the mean order size and the number of SKUs per order are this high, it could be argued that allocation based on sales *frequency* might be an improvement over allocation based on sales *volume*. The *frequency* that an SKU is ordered is defined as the number of orders that contain this SKU. When analyzing the data, a strong relationship between the sales volume and the frequency is found. Figure 9 shows the positive correlation between order *frequency* and *volume*.

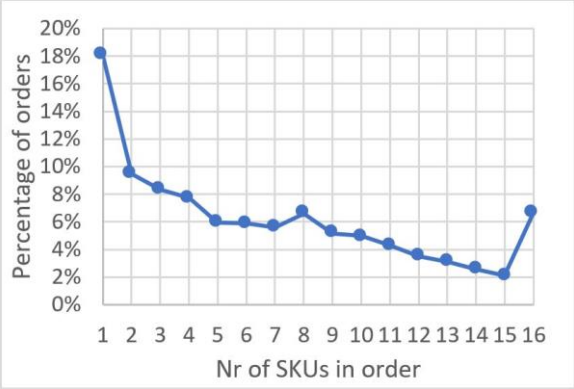

Figure 8: Number of SKUs in orders

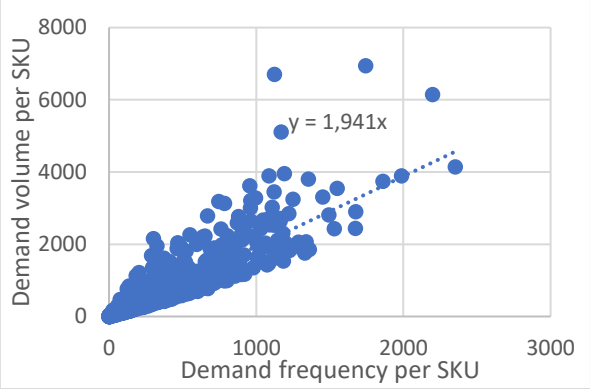

Figure 9: Correlation between sales frequency and volume

A strong relationship between the number of units and the number of order lines can be found when analyzing the order statistics. One order line is equivalent to one SKU in the order. It states the SKU ID and the volume the ordered volume of that SKU. Figure 10 shows the positive relation. It indicates that when the number of units in an order increases, the number of different SKUs also increases. The number of order units however is not dependent on the number of order lines, as visible in Figure 11. The majority of orders consist of 16 units. When 16 units are ordered, the mean number of order lines is still large, namely 6.11.

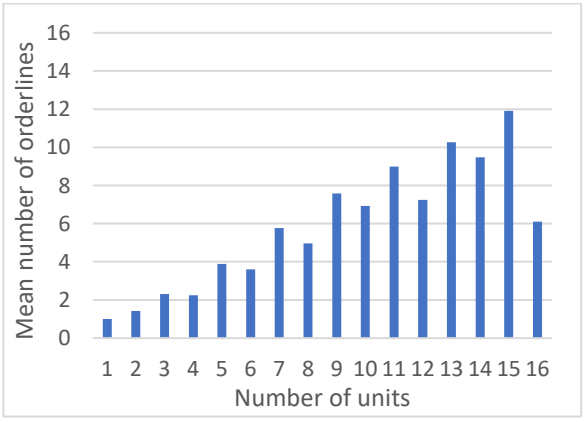

Figure 10: Mean number of order lines per number of units

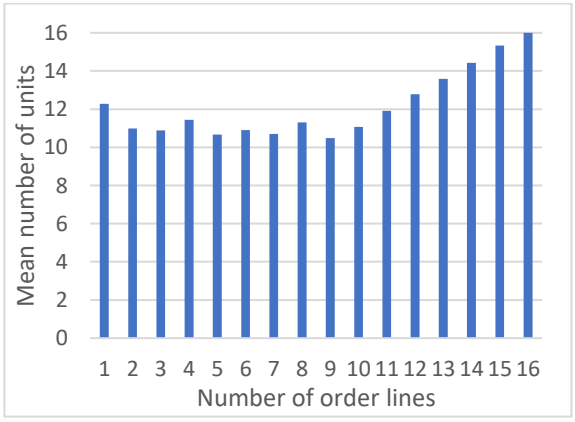

Figure 11: Mean number of units per number of order lines

Product correlation

The characteristics of the SKUs in the assortment are such that it is likely that certain SKUs have a significant probability of being ordered together. To check if strong correlations between SKUs exist,

correlation analysis has been performed that shows the SKUs that are likely to be ordered together. Around 200 relations have been found in which an SKU has a probability larger than 0.2 of being ordered when its correlated SKU is ordered. Three examples are shown in Table 1. The table shows one pair of SKUs for each example. The frequencies in which the SKUs are ordered per period are stated. The minimal frequency that an SKU must be picked per period to be allocated to the front section is given in the second column (frequency threshold). If the SKU is allocated to the front section of the warehouse in a period, this cell is colored grey. The first example shows the most frequently ordered pair of SKUs. The second example shows the most frequently ordered SKU with the SKU that has the strongest correlation with this SKU. The third example shows the pair of SKUs with the strongest correlation. For most pairs with strong correlation, like example 1 and 3, we see that both SKUs are placed in the same zone of the warehouse. Employing methods that use correlations to change allocation decisions, therefore, does not affect these SKUs. The table indicates that for example 2, using correlations to reallocate SKU δ to the front section of the warehouse could be beneficial.

		Example 1		Example 2		Example 3	
Period	**Frequency threshold**	**SKU α**	**SKU β**	**SKU γ**	**SKU δ**	**SKU ε**	**SKU ζ**
0	18	46	21	42	6	66	44
1	24	65	29	39	13	75	68
2	36	119	42	84	7	92	73
3	67	295	183	220	91	308	119
4	55	563	351	470	323	2	0
5	37	338	259	133	22	0	0
6	29	307	224	104	13	0	0
7	18	89	41	119	11	0	0
8	13	67	23	150	16	0	0
9	16	70	25	135	37	0	0
10	21	104	44	235	52	0	0
11	17	137	113	623	277	0	0

Table 1: Order frequency of SKU pairs and their placement in the warehouse indicated by color

Order batching

The current order batching method that is explained in Section 2.2.3 is suboptimal. Only 46 out of 1017 batches made in the period between 3-2-2020 and 13-3-2020 contain only orders with locations that are all in the front section while 35.88% of all orders contained only SKUs that are located in the front section. It is therefore assumed that batching can be done more efficiently to minimize the travel distance and time of the pickers.

Currently, the batches are made to minimize the number of different locations that a picker has to visit. The start-up time that a picker has each time he/she visits a new location is only a couple of seconds. The trade-off between walking through the back section of the warehouse and visiting more locations is investigated. The stop and search time of pickers in this warehouse is negligible since a scanner shows the picker the location of the next SKU. The picker does however need to scan the location and the article to check if both are correct. This takes approximately 6 seconds. With a cart, the velocity of a picker is measured to be 0.79 m/s. The distance from the I/O point to the end of the aisle is approximately 80 meters long and the cross-aisle is located after 50 meters. The extra distance that pickers walk to visit the back section of the warehouse is therefore 60 meters. This will take pickers 76 seconds extra. 12.7 extra location visits can be made at the same time that pickers would need to walk through the back section of the warehouse. Using the current batching method 33.98 different

SKUs are in one batch on average. When two order pools are made based on zones in which SKUs of orders are located, and batching is done within those pools using the same procedure, 34.46 different SKUs are in one batch on average. This is a minimal difference in stopping time per batch. Therefore, minimizing the picking tours through the back section of the warehouse should be prioritized over minimizing the number of stops in a picking tour.

2.4 Performance issue with the current system

The performance issue with the current system is that many picking routes go through the back section of the warehouse (the area between the middle cross aisle and the end of the aisles) since SKUs must be picked here. In 2020, 95% of the picking routes traveled through the back section of the warehouse. This can be improved by creating a new and better batching procedure. To improve the batching, the allocation of SKUs in the warehouse must be such that the batching procedure can use this allocation efficiently. The data analysis in the previous section of the report shows us that storage allocation and batching can be improved.

Storage allocation

The first issue with the current allocation system is that demand *volume* is being used to allocate SKUs to warehouse sections. As stated earlier, the mean number of different SKUs, in this case, is much higher as in typical warehouses. Since the number of units that are picked per SKU does not influence the picker travel distance, using the sales *volume* for allocation seems undesirable. Therefore, this thesis will investigate and employ methods that use the demand *frequency* to allocate the SKUs.

The second issue is that a forecast is made monthly in the current system to allocate SKUs. The data analysis in the previous sections shows that the data is unpredictable and that forecasting data to allocate SKUs is not optimal. Using the data used to visualize the long tail to identify the most popular SKUs (top 200) and using a method that allocates these SKUs to the front section and seek the best allocation for the other SKUs might improve allocation.

The third and final issue with the allocation method is that the methods that are currently used do not consider any possible correlation between SKUs. SKUs exist that are allocated to the back section of the warehouse with strong correlations to SKUs in the front section of the warehouse. Creating a model that allocates some of these SKUs to the front section of the warehouse at the expense of other SKUs might improve allocation since more orders could consist of only SKUs that are located in the front zone only.

Order batching

The method that is currently used to batch orders does not minimize the travel distance of pickers. It minimizes the number of unique locations that have to be visited by pickers. Improvements can be made by batching all orders that only contain SKUs located in the front section of the warehouse before batching the other orders. By creating batches that only contain SKUs located in the front section, pickers will not have to travel through the back section of the warehouse. This will improve the total picking distance. Therefore, minimizing the number of batches that have to visit the back section should be prioritized over minimizing location visits in a batch in this case.

The company wants to improve the process without making large investments. Therefore, this thesis will investigate methods to improve the allocation and batching methods by solving the SLAP and OBP with the restrictions and assumptions specific to this setting. This research tries to find a method to improve order picking efficiency by decreasing picker travel distance.

3. Problem formulation

Productivity at e-commerce warehouses constantly needs to be improved in order to be able to handle the complex picking process without large investments. The current way of storage location assignment and order batching indicates opportunities for improvements.

3.1 Problem statement

In this thesis, we consider a picker-to-parts e-commerce warehouse. In this system, order pickers travel through aisles to collect and retrieve items that are requested from storage locations within these aisles to fulfill customer orders (Dallari et al., 2009).

The performance issues, indicated through the data analysis in the previous section of this report, of the current way of allocating SKUs to storage locations and order batching ask for a new logic such that the current order picking travel distance will be reduced.

3.2 Research objective

The goal is to fulfill the customer orders with minimized travel distance. Even though this research aims to reach this goal primarily through optimization of the storage location assignment, a significant part of the thesis deals with the batching of orders. The objective of this research will be defined as: "Improving order picking efficiency by optimizing storage allocation and batching methods".

3.3 Research questions

The main research question that will be researched in this thesis is:

How to decrease order picking travel distance by solving the storage location assignment problem and order batching problem?

Sub-questions are formed to support answering the main question. These sub-questions are stated below:

- *What types of storage policies are available?*
- *What order batching method should be used given restrictions and assumptions in this case?*
    - *How does the batching policy perform compared to the current method?*
- *What is the best (combination of) storage allocation method(s) to use given the batching method in this situation?*
    - *What are the optimal parameters for the chosen storage policy?*
- *What is the order picking travel distance in the new situation?*

Extensive literature research is performed to find the answer to the first sub-question. The second question will be answered based on testing of the chosen batching policy. Given the chosen batching policy, different allocation methods are tested and analyzed to find the best performing method in this situation. The final question will be answered analytically by calculating travel distances using the case study.

4. Literature review

The study presented in this report is related to warehousing literature. More specifically, the study is related to e-commerce order fulfillment and investigates literature that answers the question: “How does one improve order picking efficiency by solving the storage location assignment problem and using an efficient batching method?”. Literature is investigated on the topics of e-commerce warehousing, storage location assignment, and order batching. To support answering the research questions in Section 3.3.

4.1 Characteristics of e-commerce warehousing

The scope of this project is mostly that of e-commerce distribution centers. E-commerce DCs typically have the following characteristics (Boysen, de Koster, et al., 2019):

- (1) Small orders are received. Bartholdi & Hankman (2011) state that orders usually consist of 1-3 items. Amazon warehouses in Germany, for example, receive orders that average around 1.6 items per order (Boysen, Stephan, et al., 2019).
- (2) Large assortments. E-commerce retailers can offer large assortments to their customers because they are not restricted by the storage space by which traditional retailers (brick and mortar stores) are restricted. Therefore, websites can offer significantly more products, which might have a small demand.
- (3) Tight delivery schedules. Most online retailers promise their customers next-day or even same-day delivery to compete within the market. Promises such as these increased stress on the operations performed within the warehouse. Warehousing activities essential for order fulfillment in e-commerce, therefore, have to be performed under more time pressure than traditional warehouses (Yaman et al., 2012).
- (4) High variation in workload. Many online retailers face highly volatile demand. Depending on the product which is offered this has several reasons, e.g. seasonality or promotions. In combination with the unpredictable demand which is often faced in e-commerce, this asks for scalable warehouse capacities that can flexibly be adjusted to varying demands and workloads. (Laudon & Traver, 2016)

Boysen et al. (2019) investigate warehouse systems and their suitability for e-commerce. In this research, the authors judge the suitability of the systems based on the four characteristics that are mentioned above. Yang et al. (2020) mention another characteristic of e-commerce warehousing: (5) the limited space of warehouses. This characteristic can be coupled with the fact that e-commerce assortments are large (2). Therefore, the availability of space in e-commerce might be more of a limitation than for other types of warehouses.

4.2 Order batching literature

Many different order-picking policies exist in literature. One of the picking policies that fit the best for e-commerce warehouses and that is investigated in this report, is batch picking. Warehouses that use the batch picking policy, combine (batch) multiple orders together. A picker then picks all the items in one of the created batches. Orders cannot be split between pickers and a picker can sort the items while picking (Yoon & Sharp, 1996). By operating the batch picking policy, pickers can pick multiple orders at a time. This minimizes travel time for the picker. In this policy, the process for order pickers is the same as for strict order picking. A picker travels to all pick locations on a list and picks every item until all SKUs are picked. The picker then travels back to the I/O point and obtains a new picklist. This is done until all orders are picked (Petersen, 2009). The disadvantages of this policy are; the loss of integrity, an increase in the potential for errors, and space for consolidation if the items are sorted downstream. When pickers sort the items while picking, this can be avoided together with the loss of

integrity. Therefore, effective batch picking is realized when a balance is found between travel savings and the cost of sorting and errors (Frazelle & Apple, 1994).

The Order Batching Problem (OBP) is not to be confused with the concept of batch picking, which is described above. The OBP aims to optimize the method in which orders are combined into batches such that picking route distances are minimized. OBP is generally categorized as either static or dynamic (Aboelfotoh et al., 2019). In a static OBP, the orders are received before batching starts. In a dynamic OBP, the orders arrive dynamically over time. One batch construction that is the simplest, is the FCFS (first come first serve) method. Using this method, orders are assigned to batches based on the order in which they arrive. This method, however, usually results in longer picking tours compared to other methods. Recent studies address the joint optimization of order batching and routing.

Few studies have been conducted on the application of mathematical models to batch orders because these models can only be applied to situations in which the OBP has to be solved with a small number of orders. Researches by Bozer & Kile (2008) and Gademann & van de Velde (2005) using mixed-integer programming and a branch-and-price algorithm respectively to solve the OBP for up to 32 customer orders and found that increases in the number of orders, the batch size, and the number of aisles made the problem harder to solve.

Numerous studies have been conducted on developing and applying metaheuristics. These can be used with larger-scale problems to solve the OBP. Hsu et al (2005) proposed a batching method using a genetic algorithm that was outperformed by the Artificial Bee Colony algorithm by Li & Zhou (2013). Many studies such as Henn & Wäscher (2012) and Žulj et al. (2018) apply tabu search-based methodologies to solve the OBP. Menéndez et al. (2017) and Pinto & Nagano (2019) proposed heuristics based on variable neighborhood search strategies.

4.3 Picker routing problem literature

Order-picking is the most labor-intensive activity in most warehouses. Bartholdi & Hackman (2011) show that order picking accounts for most of the warehouse operating costs. 55% of the total operating cost is accounted for by order-picking and more than half of the order picking time is consumed by the travel time of the pickers. Therefore, the reduction of travel time can greatly reduce operating costs within warehouses. Storage costs can be reduced with small reductions of distances traveled by pickers. This can be achieved through efficient picker routing planning. The efficient planning of these routes seeks to minimize the total distance traveled given the locations that have to be visited to pick the orders (Scholz et al., 2016). Routing policies, therefore, determine the sequence in which SKUs are to be picked by a picker. Different approaches are used to tackle the routing problem; routing strategies or policies, heuristics, metaheuristics, and optimal procedures (Cano et al., 2017). In this section, those approaches will shortly be summarized.

The most common routing policies that are used in literature are the transversal (or s-shape), return, midpoint, largest gap, and composite policies. Figure 12 illustrates these policies.

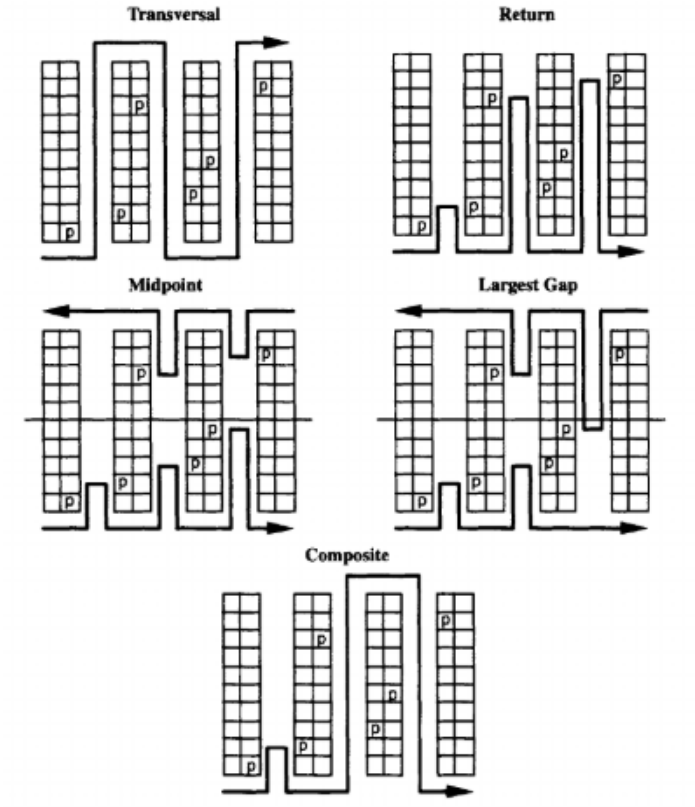

Figure 12: Common routing policies (Petersen & Schmenner, 1999)

Different heuristic methods to solve the routing problem are proposed in literature. Won & Olafsson (2005) propose a heuristic called the 2-opt heuristic. Other heuristics that are proposed in literature are; the Largest Gap with Simulated Annealing (Ho et al., 2008), Nearest Neighbor and savings heuristics (e.g. Kulak et al., 2012), traveling salesman, and weight heuristics (Choy et al., 2014; Theys et al., 2010). Metaheuristics that are developed to find the most efficient traveling path are; genetic algorithms (Azadnia et al., 2013), ant colony optimization (J. Li et al., 2017), particle swarm optimization (Gómez-Montoya et al., 2016), and tabu search (Cortés et al., 2017). Optimal procedures include algorithms to find the optimal picking route using the graph theory combined with dynamic programming (Ratliff & Rosenthal, 1983), as well as a solution through the graph system (Scholz et al., 2016) and a branch-and-cut algorithm (Valle et al., 2016).

Heuristic strategies are more straightforward and easier to follow than the other solutions to routing problems (Petersen & Schmenner, 1999). Petersen (1999) states that optimal solutions can create confusion for the picking operators. Hall (1993) describes that optimal methods are usually a combination of s-shape and largest gap heuristic strategies and that the heuristic methods can, therefore, develop near-optimal routes with less confusion. For these reasons, warehouses often choose to use heuristics rather than optimal procedures.

4.4 Storage Location Assignment Problem (SLAP) literature

Storage is an important factor with which is dealt with by the operating policy in warehouses. As explained above, 55% of the order picking time is consumed by the travel time of the order-pickers. Travel can be reduced by effective storage allocation. SLAP is a well-known concept in warehousing. SLAP focuses on the allocation of SKUs to specific locations in the warehouse and also considers combining the most similar orders in a batch given the assignment of SKUs. Characteristics such as frequency, size, weight, and supplier are often used when assigning items to locations or zones in the warehouse. In order to propose the solution method that is mentioned later in this report, the basics

of storage location assignment are introduced in this section. Deciding on the storage location assignment policy is an important factor in deciding the allocation of SKUs to exact locations.

Different types of storage assignment policies have been identified in literature. Basic policies include random storage, closest open location, dedicated storage, and class-based storage. Using the random storage policy, items are stored in random locations and do not follow any logic as to where the most efficient location is to store the SKU. Random storage causes high storage utilization in the warehouse and low order-picking efficiency (de Koster et al., 2007). With the closest open storage policy, items are stored in the closest open location from the I/O (input/output) point of the warehouse (Park & Lee, 2007). The advantage of this policy in comparison to random storage is the low warehouse utilization. When operating under a dedicated storage policy, specific storage locations are assigned to each SKU to be stored (M. K. Lee & Elsayed, 2005). Family grouping, volume-based storage, affinity-based storage (Bartholdi & Hackman, 2011) and full turnover-based storage are examples of different types of dedicated storage that exist. The turnover-based policy is often used in practice. This policy is expected to minimize the time required to store and retrieve products. Full turnover-based storage, unlike the other types of dedicated storage, assigns locations to items based on historical data. It examines the turnover of each SKU and aims to store the SKU with the highest activity ratio closest to the I/O point within the warehouse. Due to the minimization of storage and retrieval time, this storage policy might be well suited to cope with the time pressure of e-commerce. Class-based storage is a storage policy in which items are divided into a number of storage classes. Within those classes, the storage locations are randomly assigned to each item (Petersen et al., 2004). It outperforms random storage and approaches volume-based storage in order-picker travel time. It also outperforms dedicated storage in the total cost of order picking. A form of class-based storage is ABC storage. Le Duc & de Koster (2005) explain that in this storage policy, items are subdivided into three categories, generally based on the nature and size of the item. Fast-moving items that are small in size are categorized as an A-item and large items that are slow-movers are categorized as C-items. A-items are placed closer to the depot than C-items. Category B items are in between the other two categories, both in characteristics and placement. With smaller assortments, one can create clear and static categories that do not change much over time. This storage policy decreases traveling distance of order pickers, supporting the tight delivery schedules in e-commerce warehouses.

A more complex storage policy is dynamic storage. This storage policy aims to enhance the order picking process performance by dynamically storing only those SKUs needed for the current order batch in the pick area, thereby reducing travel time for this process. The other SKUs are stored in the reserve area (Yu & De Koster, 2010). Yu & De Koster (2010) show that using this system, higher throughput can be realized than by using a conventional order picking system since only a small portion of the total SKUs are stored in the forward pick area. This storage policy is a system that has proven to work in several e-commerce warehouses (e.g. Amazon Germany).

Classic storage strategies perform well when only one SKU is picked in a picking tour. In e-commerce, as explained, the number of picks per order is low. Thus, these policies might be suitable for an e-commerce warehouse. However, warehouses that receive more orders with multiple SKUs that have to be picked ask for a more complex storage assignment method. Identifying correlated SKUs in the assortment and placing these items closer to each other can improve efficiency (Zhang et al., 2019). Frazelle & Sharp, (1989) calculate the pairwise correlation between item pairs using historical order data and define the SLAP as an integer programming problem, which is solved using a two-stage heuristic. Liu (2004) proposes the order-item-quantity rule to measure similarity between item pairs. These items are then located considering the used routing strategy. This method is commonly employed in practice. Many studies exist that use a clustering method to create groups of items that

are stored together. Lee (1992) and Hua & Zhou (2008) propose new clustering strategies that locate items following (space-)filling curves. Chuang et al. (2012) improve the family grouping policy by measure pairwise item association and clustering items. They use mixed integer programming for both the item clustering and assigning the clusters to locations in the warehouse. Jane & Laih (2005) propose a clustering strategy for zone picking systems and develop a heuristic that assigns correlated items to different zones using clustering. In recent years, different approaches that employ correlation have been proposed in literature. These approaches do not use clustering but assign products otherwise. Pang & Chan (2017) analyze association relationships between different items in customer orders. They consider put-away cost and use a data mining-based algorithm to solve the SLAP. Both item demand and correlation are considered by (Glock & Grosse, 2012).

Some recent studies also use some kind of local search algorithm to solve SLAP. Neighborhood search algorithms have been used by Hansen et al. (2020) to solve the SLAP and by Yang et al. (2015) to solve optimize the joint problem of storage location assignment and storage/retrieval schedule. The research in this thesis also uses a local search algorithm to solve the SLAP.

4.5 Gaps in literature

In the research by Boysen et al. (2019) suitability of warehouse systems in e-commerce is judged based on the four characteristics that e-commerce DCs usually have. The authors do not focus on warehouses that do not have (all of) those characteristics. Some e-commerce warehouses, such as online grocery retailers, for example, do not have the characteristic that most small orders of 1-3 items are received. Other warehouses, such as the e-commerce warehouses that execute operations for one-product or one-page webshops, do not have the characteristics of a large assortment. No further research has been conducted in warehouse systems that fit for warehouses that do not have those four characteristics or that have other constraints. The research presented in this thesis will add value to literature by researching allocation and batching methods for a warehouse that has a larger mean order size than typical e-commerce warehouses. This research solves the SLAP in a warehouse with two picking aisles and three cross aisles, creating two zones in which a variant of the turnover-based storage policy is applied. Both demand and correlation between orders are used to solve the allocation problem. Literature also often uses data of a period to solve SLAP for the same period in which customer orders arrive. In practice, SLAP has to be solved with historical data before new customer orders arrive. These new customer orders are then picked using the proposed location assignment found by solving SLAP. This research uses this practical approach to solve SLAP instead of the approach that is often used in literature. The OBP in this type of warehouse is not investigated in literature. Simple methods are used in this thesis to solve the OBP using the layout of the warehouse.

5. Conceptual design

In this section of the report, the conceptual model that will be used to design a new batching and allocation system is introduced. In order to create such a model, the objective, the performance indicators, functional requirements, assumptions, and constraints need to be specified. After defining these, the conceptual model will be explained.

5.1 Objective and performance indicators

Before designing a model and solution methods, the objective of the model must be clear. The objective of this research is to minimize picker travel distance. The model needs to allocate SKUs in a warehouse such that picking tour distances to collect customer orders are minimized. Batches of maximum of six orders must be created that support the allocation decisions by efficiently combining customer orders such that order pickers can pick all batches created with minimized travel distance.

To determine the effect of the proposed methods, one ultimate performance indicator is used: the travel distance per order. After both allocation and batching decisions, the total distance that is traveled to pick the orders will be calculated to determine the performance of the methods. The routing distance calculations will be executed based on the expected distances that order pickers travel in the aisles. The two routes that are traveled are deterministic. The total distance, however, is dependent on the mix between A-batches and AB-Batches. A-batches are batches for which the picker only travels through the front zone of the warehouse. The orders in those batches only contain SKUs that are located in this front zone. These orders are referred to as A-orders in this research. AB-batches are batches of which the orders contain SKUs located in the front and back section of the warehouse. Those orders are referred to as AB-orders. To pick an AB-batch, the picker walks a longer route through the back section of the warehouse. In the detailed design section of the report, more explanation on the calculations is given and the equations to calculate the distances are defined.

Another KPI used in this report is the number of stops per batch. Pickers travel through aisles to pick a batch of orders. Depending on the number of different SKUs that are in this batch, the pickers must make a number of stops at a picking location. Minimizing the number of stops will decrease picker travel time and therefore reduce picking costs.

In this research, reallocation costs are not quantitively considered. Because of the continuously changing assortment, it is unavoidable to reallocate SKUs at least once per month. In this research, it is assumed that allocation decisions are made every 28 days and are effective during this period. Actively reallocating SKUs more than once in this time period will lead to high reallocation costs which are not desirable. If SKUs would be reallocated weekly, for example, the allocation costs would be extremely high since a large number of case packs would have to be reallocated because of the large difference in weekly demand.

5.2 Decision hierarchy

To get to the objective described above, the model in this research uses two steps. First batching decisions are made, and second allocation decisions are made. According to Bertrand et al. (2016), the time period over which a higher-level decision is effective is not smaller than the time period over which a lower-level decision is effective. Figure 13 illustrates the connections between decision layers and the system behavior in this research.

The period of time over which allocation decisions are effective is larger than the period of time over which batching decisions are made. The allocation policy does not change every period of 28 days. Changing allocation policies might cause major reallocations which are costly. Changing the batching

policy, however, can be done daily. Therefore, the batching policy that is used is seen as a lower-level decision.

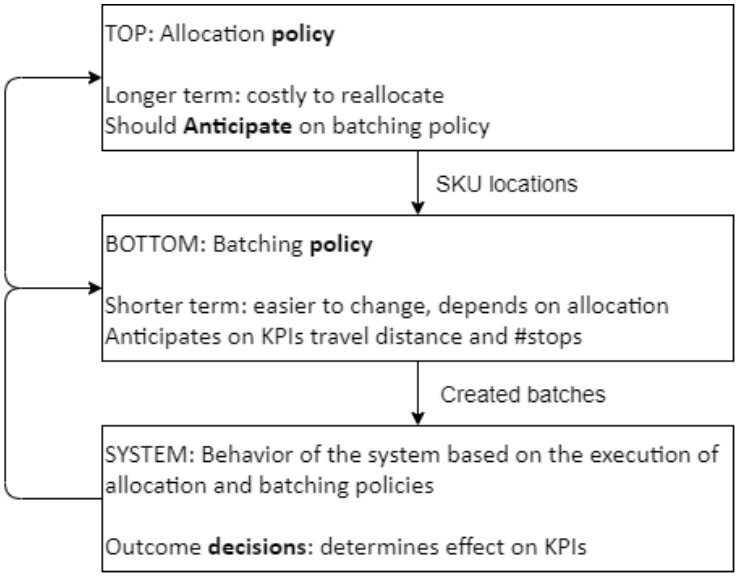

Figure 13: Decision hierarchy and data needed per layer

Using the performance indicators described earlier, the performance of the allocation methods can only be measured after batching decisions are applied and a dataset is analyzed using these decisions. In this research, multiple allocation methods are analyzed. Each time that different methods of allocation are analyzed, the OBP has to be rerun to find the performance of the allocation method. The allocation model outputs where SKUs will be located in the warehouse. Therefore, after the allocation decisions, it is known which orders in a dataset are categorized as an A-order and which as AB-orders. This data is needed to solve the OBP. For the distance calculations, the OBP has to be solved in order to find the mix between A-batches and AB-batches.

5.3 Model assumptions and constraints

Several assumptions and constraints exist in this research to define the scope of the model. The assumptions and constraints that must be accounted for when creating the model are listed below:

Assumptions

- Orders are received one day and picked the next day
- Every day one pickup moment exists at the end of the day at which all batches are picked up
- All SKUs can be stored in any storage location in the warehouse
- It is assumed that there are enough forward locations in the warehouse for all SKUs
- All orders for one day are received before batches are made
- New SKUs in the assortment will always be allocated in the back section of the warehouse
- SKUs are reallocated after periods of 28 days
- Allocations are done at the end of a period using data of that period
- The reallocation of SKUs is done actively. SKUs do not stay in the same location until the location is picked empty but are actively moved to their new location
- An order can be batched with all remaining orders

Constraints

- Layout constraint: the layout consists of 2 aisles with 3 cross aisles as indicated in Figure 14
- Routing constraint: the two aisles allow just two different picking routes. These are indicated in Figure 14
- Batching constraint: Batches consist of maximum 6 customer orders
- Order constraint: each order consist of maximum 16 units

5.4 Layout and Routing

Figure 14 illustrates the warehouse layout. The warehouse has two pick-aisles and three cross aisles which provides access from the first pick aisle to the second. Two cross aisles are located at the top and bottom of the aisles. The middle cross-aisle is indicated in Figure 14. The aisle width of the warehouse could be a factor that has to be taken into account. For this research, the aisle width is assumed to be negligible, i.e. it is assumed that the aisles are narrow enough for the order pickers to be able to pick items from both sides of an aisle without moving sideways. Hall (1993) states that for aisle width below $2.71 \times expected\ distance\ per\ pick\ per\ unit\ aisle\ lenght$ the width of an aisle can be considered negligible, i.e. the picker can collect from both sides of the aisle. Therefore, order pickers travel through the middle of the aisle and backtracking is not considered. Two different routes that can be travelled by the order pickers. The travel routes of the pickers are directed and essentially determined by the S-shape routing procedure as described by Dijkstra & Roodbergen (2017). The cross-aisle can be used to travel from the first to the second pick-aisle when no SKUs have to be picked in zone B. The two different routes that can be traveled by pickers are illustrated in Figure 14. Regardless of which route is traveled, the pickers always travel through all products located in the front section of the warehouse.

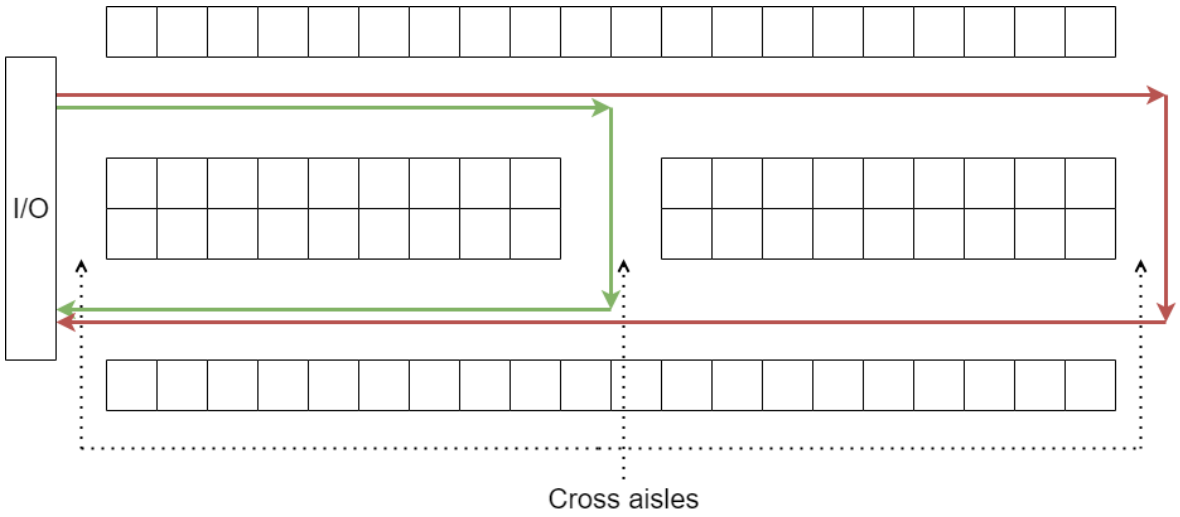

Figure 14: Warehouse layout and picking routes

5.5 OBP

In the considered situation, orders are batched for order pickers to pick during a picking tour. The batching of the orders has an impact on the total travel distance of the order pickers. Optimal batching of orders is, therefore, necessary to fulfill the orders with minimized travel distance. Since only two different routes can be traveled in the problem instance that is focused on, route optimization can be simply achieved. Given all assumptions mentioned earlier, together with the routing and layout of the warehouse, a new batching method can be created that minimizes the number of picking routes through the back section of the warehouse, thus reducing the total picking travel distance.

The new batching method groups all orders into so-called A-orders and AB-orders. A-orders are orders containing only items that are allocated in the front section of the warehouse. AB-orders contain items that are in both the front and back sections of the warehouse. A decision has to be made on what happens when there are not enough orders left in the A-orders pool to create a full batch (< 6 orders). One can either create an incomplete A-batch or add the remaining orders to the AB-pool. The first method could, worst case, cause a picker to walk the picking route for only one order while that remaining A-orders could have been added to the last AB-batch that consists of less than six orders. The second method, however, could cause a picker to walk an extra route for an AB-batch when the number of remaining A-orders added to the remainder of AB-orders after creating AB-batches is more than six. The second method is used in this thesis since this method minimizes the total number of batches.

After grouping the orders in order pools, the batching method first batches all A-orders. These batches are referred to as A-batches. After creating the A-batches, the method creates AB-batches out of the remaining orders. Batches within those order pools are created using the batching logic that is currently used by the company. This method is used to minimize the number of stops made per batch that must be picked. The method uses the Jaccard coefficient to calculate the similarity between order pairs. The order-pair with the highest coefficient is combined and new Jaccard coefficients are calculated for each order with this order pair. Again, the order with the highest coefficient is added to the pair. This process continues until six orders are in one batch. A new starting pair will then be created and the entire process is repeated. Using the new method, many order picking routes will only travel the shortest of the two routes explained in the previous section of this report.

5.6 SLAP

The batching policy asks for an allocation model such that pickers can pick orders with minimized travel distance. Different allocation methods will be analyzed to choose the most suitable method in the situation that is described in this thesis. An optimization problem will be formulated for a certain time period for which the allocation problem must be solved.

Policies

This research considers a class-based system with two different classes to tackle the SLAP. In literature, the most common method for class-based storage in e-commerce warehousing is the class-based turnover policy. As explained earlier, the mean order size, in this case, is larger than in typical warehouses. Because of this, the turnover-based (TOS) policy might not be most efficient and another method that allocates SKUs based on the *frequency* of ordering (instead of the total order *volume*) has been investigated to solve the SLAP. The method is referred to as Frequency of Ordering (FoO) in this thesis. Company data will be used to check whether picking travel distance will be reduced using the FoO method. In the case that only one SKU is picked per picking tour, allocating SKUs strictly by frequency should outperform allocation on-demand *volume* when minimizing picking travel distance. In this specific case, however, many SKUs are ordered together in one order. Because of this, the probabilities that SKUs are ordered together are of importance. The allocation of SKUs is not only dependent on the frequency that the product is picked, but also on the probabilities that it is ordered with other SKUs. To attempt to solve this problem, a greedy construction algorithm is created.

Greedy construction algorithm

In the situation at K+N, the SKUs do not correlate one-to-one and all items have a conditional probability of occurring in an order given that other SKUs are in this order. The SLAP that is considered when using the chances that SKUs are ordered together is a quadratic assignment problem. This

problem is NP-hard and cannot be solved exactly in polynomial time. A greedy construction algorithm is used to attempt to solve this problem. This algorithm uses a correlation analysis to allocate some SKUs that are initially located in the back section of the warehouse, to the front section if these SKUs are ordered frequently and have a high correlation with (multiple) SKUs located in the front section. This SKU is allocated to the front section at the expense of another SKU. Section 2.3 shows that demand cannot be forecasted accurately in this situation. Next to the inaccurate demand forecast, the section also shows that correlations seem to be unpredictable. Reallocating SKUs based on the historical probability of being ordered with other SKUs does not guarantee that this SKU also has this probability in the next time period. Therefore, it is possible that using this greedy construction algorithm, might not lead to better performance than the allocation policies described above.

Hill-climbing local search algorithm

When sequencing SKUs on frequency, the frequencies of the SKUs in the area around the capacity limit of the front zone do not differ much. A local search algorithm is therefore used to find the swaps between SKUs in the front and back section that actually improve the allocation. The local search algorithm in this thesis uses an initial allocation that is made using the best performing policy of the policies described above (FoO or TOS). It then swaps one SKU in the front section of the warehouse with another SKU in the back section. It checks whether this swap results in a better allocation in terms of picker distance after batching. If the swap does not improve the performance of picking, including batching, the swap will be reversed. If the swap does improve the performance, it will be saved. This algorithm is therefore seen as a hill-climbing local search algorithm that is only accepting improvements. The allocation made using this algorithm will then be used to pick the orders that are received in the next time period. Many iterations of this algorithm can be run to keep improving allocation. Running local search algorithms, however, can be very time-consuming. Therefore, an adequate number of iterations should be chosen. Alongside the number of iterations, the area in which the swaps will be made has to be determined. Since the data analysis in Section 2.3 showed that the top 200 SKUs do not alternate much. A window of 400 SKUs around the capacity of the front section of the warehouse (which is 400 SKUs). Thus, with SKUs indexed on the frequency of ordering, SKUs 201-400 will be swapped with SKUs 401-600. In Section 6 of this report, the hill-climbing local search algorithm will be explained in detail.

6. Detailed design

In this section of the report the conceptual model will be explained in detail and the solution approaches are elaborated. Formulas and algorithms are given that are used to solve the model.

6.1 Parameters and decision variables

Below, the parameters and main variables for this problem are described. These will be used in the remainder of this research.

Parameters:	
$\boldsymbol{i}$	$\in N$. An index to indicate a specific SKU
$\boldsymbol{N}$	The set of all SKUs in the assortment
$\boldsymbol{o}$	$\in O$. An index to indicate a specific order
$\boldsymbol{O}$	The set of all orders
$\boldsymbol{b}$	$\in B$. An index to indicate a specific batch
$\boldsymbol{B}$	The powerset of all batches. This is the largest number of potential batches
$\boldsymbol{F_i}$	Frequency of ordering SKU i
$\boldsymbol{V_i}$	Order volume of SKU i
$\boldsymbol{C}$	The capacity in number of SKUs in the front section of the warehouse
α	The weighting factor of the correlation
$\boldsymbol{P_{ij}}$	The chance that SKU i and j are ordered together
$\boldsymbol{a_i}$	Value to index SKUs based on both demand and correlation
$\boldsymbol{M}$	The neighborhood of SKUs from the border capacity
$\boldsymbol{o_{ij}}$	An order containing both SKUs i and j
$\lvert\boldsymbol{V_o}\rvert$	Number of orders
$\lvert\boldsymbol{V_{o_{ij}}}\rvert$	Number of orders that contain SKUs i and j
$\lvert\boldsymbol{V_{o_i}}\rvert$	Number of orders that contain SKU i
$\boldsymbol{C_{back}}$	Cost of visiting the back section of the warehouse
$\boldsymbol{C_{stop}}$	Cost of visiting an extra location
$\boldsymbol{j_{XY}}$	Jaccard coefficient of order pair XY
$\lvert\boldsymbol{o_a}\rvert$	Number of orders in A-order pool
$\lvert\boldsymbol{o_{ab}}\rvert$	Number of orders in AB-order pool
$\boldsymbol{r_a}$	Distance traveled for one A-batch
$\boldsymbol{r_b}$	Distance traveled for one AB-batch
$\lvert\boldsymbol{q_a}\rvert$	Number of A-batches
$\lvert\boldsymbol{q_b}\rvert$	Number of AB-batches
$\boldsymbol{D}$	Total distance traveled
$\boldsymbol{MD}$	Mean distance traveled per order
$\boldsymbol{A_{io}}$	$\in \{0,1\}$. SKU i is part of order o (1) or not (0);
Decision variables:	
$\boldsymbol{x_i} \in \{\boldsymbol{0},\boldsymbol{1}\}$	SKU i is located in the front section (1) or the back (0);
$\boldsymbol{y_{ob}} \in \{\boldsymbol{0},\boldsymbol{1}\}$	Order o is assigned to batch b (1) or not (0);
$\boldsymbol{z_b} \in \{\boldsymbol{0},\boldsymbol{1}\}$	Batch b needs to visit the back section (1) or not (0) ;
$\boldsymbol{u_{ib}} \in \{\boldsymbol{0},\boldsymbol{1}\}$	the location of SKU i needs to be visited by batch b (1) or not (0);

Table 2: Parameters and decision variables used in this research

6.2 Performance measurement

As explained in the conceptual model, the performance of the decisions made on batching and allocation levels is measured using a performance indicator. In this report, the key performance indicator is the travel distance per order. A detailed description of how the distance per order is calculated is given in this section of the report.

We consider an order picking area as described earlier with 2 aisles each. The layout is shown in Figure 15. The depot is indicated as the I/O point in this figure. The length of the aisles are indicated by d_a. The middle cross aisle is located at distance d_m from the depot. The distance from the middle cross-aisle to the end of the picking aisles is indicated as d_e and the distance that is travelled from one aisle to the other using a cross-aisle is indicated as d_c. All distances in this report are measured in meters.

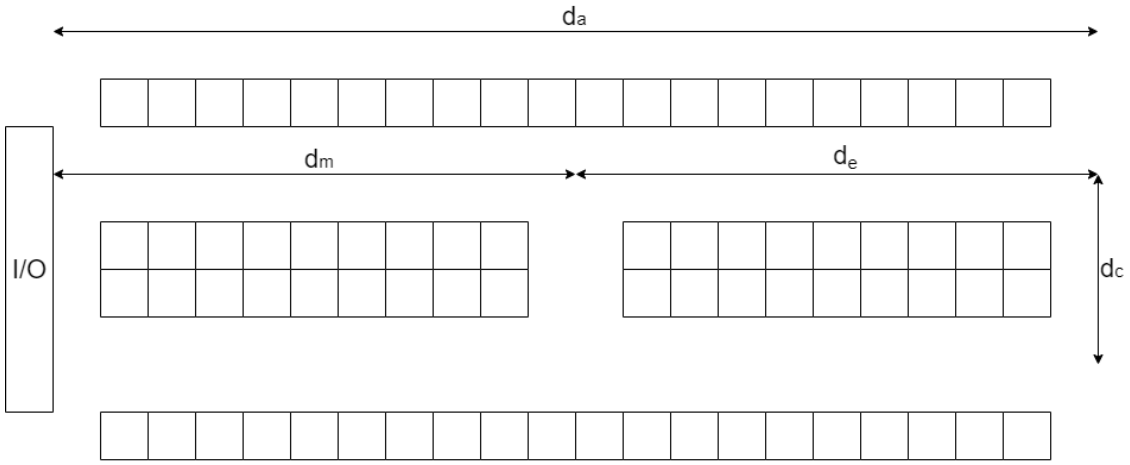

Figure 15: Warehouse layout with parameters

The routing considered is shown in Figure 14 and explained earlier as being some form of an S-shape routing strategy. The distance traveled to pick all batches in one day will be defined as D. The distance travelled from the first aisle to the second is always d_c. The distance travelled in one picking route (for one batch) is defined as either r_a or r_b which are the route lengths of a picking route through only the front section of the warehouse and a picking route through both the front and back section respectively. For every route, the distance traveled in aisle 1 is the same as the distance traveled in aisle 2, because of the routing constraints. Therefore, r_a and r_b are defined as follows:

$$r_a = 2 \times d_m + d_c$$

$$r_b = 2 \times d_e + d_c$$

D is now calculated by multiplying the number of batches for each zone with their respective route distance. Let $|q_a|$ be the number of A batches and $|q_b|$ the number of AB batches.

$$D = |q_a| \times r_a + |q_b| \times r_b$$

To compare the performance of the methods described in the previous section, the mean distance traveled per order MD is calculated. Let $|V_O|$ be the number of orders. MD is calculated as follows:

$$MD = \frac{D}{|V_O|}$$

6.3 OBP

The warehouse is split into two sections. We want to batch a set of orders in such a way that we minimize the number of batches that need to visit the back section of the warehouse. The batching procedure seeks an assignment such that the number of location visits and the number of routes through the back section of the warehouse are minimized:

$$Min\{C_{back} \sum_{b \in B} z_b + C_{stop} \sum_{i \in N} \sum_{b \in B} u_{ib}\}$$

Subject to:

$$\forall o \in O: \sum_{b \in B} y_{ob} = 1 \quad (1)$$

$$\forall b \in B: \sum_{o \in O} y_{ob} \leq 6 \quad (2)$$

$$\forall i \in N, b \in B: \sum_{o \in O} A_{io} y_{ob} \leq BigM \cdot u_{ib} \quad (3)$$

$$\forall b \in B: \sum_{i \in N} \sum_{o \in O} A_{io}(1 - x_i) y_{ob} \leq BigM \cdot z_b \quad (4)$$

Constraint (1) ensures that all orders are in exactly one batch. As explained earlier, the maximum number of orders in one batch is six. This limitation is captured in constraint (2). Constraint (3) ensures that the number of SKUs in a batch is not larger than the number of locations that have to be visited when picking this batch. Finally, if one item in a batch is located in the back section of the warehouse, the picker has to visit the back section when picking this batch. This is controlled by constraint (4).

As explained in the data analysis in Section 2.3, minimizing the number of picking routes that travel through the back section of the warehouse is prioritized over minimizing the number of stops in a picking route. The costs of visiting the back section (C_{back}) is much more than the cost of visiting an extra location (C_{stop}). To minimize the number of picking routes through the back section, indicated as $\sum_{b \in B} z_b$ in the minimization problem above, two order pools have to be created. One with only A-orders and one with AB-orders. Within those order pools, batches are made to minimize the number of unique location visits, which is indicated as $\sum_{i \in N} \sum_{b \in B} u_{ib}$. This is done using the Jaccard coefficient. The Jaccard coefficient batches orders that contain similar SKUs. The following algorithm will be used to create batches from all orders in the order pool that has been received during the previous day:

Algorithm 3: Batching procedure

Input data: Order data for one day, x_i for all $i \in N$
Result: Batches of orders ready to be picked

Add all orders $o \in O$ to A-orders pool
for $o \in O$ **do**
  for all SKU $i \in o$ **do**
    if $x_i = 0$ **then**
      Place order o to AB-orders and remove from A-orders
    else
      Leave order o in A-order pool
    end if
  end for
end for
if $|o_a| \geq 6$ **then**
  Define Jaccard coefficient j for each order pair XY in A-orders pool using: $j_{XY} = \frac{|X \cap Y|}{|X \cup Y|}$
  Create a starting pair with the combination of orders that has the highest Jaccard coefficient and remove those orders from the A-order pool
  while Number of orders in current combination < 6 **do**
    Calculate the coefficients for all remaining orders with the combination
    Add order with highest coefficient to combination and remove from pool
  end while
  Save combination and remove orders in combination from A-orders pool
else
  Add remaining A-orders to AB-orders pool
end if
if $|o_{ab}| > 6$ **then**
  Define Jaccard coefficient j for each order pair XY in AB-orders pool using:

$$j_{XY} = \frac{|X \cap Y|}{|X \cup Y|}$$

  Create a starting pair with the combination of orders that has the highest Jaccard coefficient and remove those orders from the AB-order pool
  while Number of orders in current combination < 6 **do**
    Calculate the coefficients for all remaining orders in pool with the combination
    Add order with highest coefficient to combination and remove from pool
  end while
  Save combination
else
  Batch remaining AB-orders
end if

A python script is created to use this algorithm with company order data to analyze the effect of this batching method. The Jaccard coefficient is used in this algorithm to create batches that minimize the unique locations that have to be visited by the order pickers, therefore reducing the number of times that order pickers have to stop, search, locate and start walking again. In the integrative design section, a case study is done using company order data. In that section, the performance of this batching algorithm is compared to the performance of the currently used batching method. A simple example of the algorithm is shown in Appendix A.

6.4 SLAP

The performance of the allocation and batching problem defined in the conceptual model is defined by calculating the picking travel distance. The batching procedure allows for a separate allocation problem that seeks an assignment of SKU locations such that the number of picking routes that travel through the back section of the warehouse is minimized. This, in its turn, minimizes the total picking distance traveled. In this section, the models and methods that are used to solve the SLAP are explained in detail.

Policies

Both the FoO policy and the TOS policy which are tested in this thesis can be defined as optimization problems with the assumptions and restrictions of this research. When strictly using frequency to assign SKUs to locations, the following knapsack problem should be solved:

$$Max \sum_{i=1}^{N} F_i x_i$$

Subject to:

$$\sum_{i \in N} x_i \leq C$$

$$x_i \in \{0,1\}$$

The first constraint given in this problem makes sure that the capacity of the front section is respected. The second constraint states that a product can either be allocated to the front section ($x_i = 1$) or to the back section ($x_i = 0$) of the warehouse. When the assignment of SKUs to zones is done using the TOS policy, the *frequency* of which an SKU is picked (F_i) will be replaced by the ordered *volume* (V_i) in this optimization problem.

To solve the knapsack problems, a greedy algorithm is used. A binary knapsack problem with unit-sized items can be solved exactly using a greedy algorithm. Since all items have the same weight/size, it becomes a ranking issue that is solvable exactly using a greedy method (Garey & Johnson, 1979). The greedy method simply ranks the SKUs in terms of frequency (or demand volume when using TOS policy). For the first C SKUs in this ranking, x_i will be set to 1. These items will be allocated to the front section of the warehouse.

Quadratic assignment problem

To account for the probabilities that SKUs are ordered together, another model will be is defined. In this model, the probability that two SKUs i and j are ordered together is defined as P_{ij}. This probability is conditioned between 0 and 1. Since the dimension of the probability and frequency differ, a weighting factor α is added to the problem. These additions make the problem a quadratic assignment problem (QAP), which is NP-hard. The following optimization problem is defined:

$$Max\left\{\sum_{i=1}^{N} F_i x_i + \alpha \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} P_{ij} x_i x_j\right\}$$

Subject to:

$$\sum_{i \in N} x_i \leq C$$

$$0 \leq P_{ij} \leq 1$$

$$x_i \in \{0,1\}$$

This model is a special case of the QAP. This is a linear QAP that can be reformulated as a Mixed-Integer Linear Programming (MILP) problem and solved as a traveling salesman problem (Queyranne, 1986). However, since the problem that is solved has a large number of SKUs ($N > 1400$), the problem cannot be solved exactly in polynomial time. Therefore, a greedy construction algorithm is used to attempt to solve this problem by using probabilities of SKU pairs with a high correlation of being ordered together. Next to this greedy algorithm, a hill-climbing local search algorithm is also used to attempt to find better solutions for allocation (including batching decisions). Both methods are explained below:

Greedy construction algorithm

The greedy construction algorithm utilizes the correlations of SKU pairs with a high probability of being ordered together to reallocate products. It uses a correlation coefficient to place certain SKUs in the front section of the warehouse. These were the SKUs that were placed in the back section of the warehouse and which had a strong correlation with SKUs in the front section of the warehouse. For SKUs with a support value larger than 0.004, a correlation coefficient is calculated. A support value of higher than 0.004 means that the SKU pair is present in more than 0.4% of all orders. This threshold is chosen because all item pairs that seem to truly correlate have a higher support value. Most pairs with a lower support value do not have one SKU that is ordered (frequent) enough to be allocated to the front section of the warehouse or do not have a high correlation coefficient. This coefficient is defined as the confidence; the probability that SKU j is ordered if SKU i is ordered. This coefficient is calculated as follows: $\frac{|V_{O_{ij}}|}{|V_{O_i}|}$. The confidence is used to approximate P_{ij} in the objective function above. The greedy construction algorithm utilizes these probabilities to create a correlation coefficient for a single SKU and create a new ranking problem by adding this correlation coefficient to the frequency or volume (depending on if FoO or TOS is used as the initial policy) that an SKU is ordered. A correlation matrix is created that contains all correlation coefficients of SKU pairs i and j. For the item pairs with a support value smaller than 0.004, the correlation matrix will have a value of 0. The greedy algorithm below utilizes this matrix to determine a new allocation. This greedy algorithm can be used with both the TOS and FoO policies, the greedy algorithm below uses the FoO policy. If the TOS policy is used, F_i has to be replaced by V_i in the algorithm. All correlations found in the correlation matrix can be found in Appendix B. A simple example of the greedy algorithm can be found in Appendix C.

Algorithm 1: Greedy construction algorithm

Input data: Historical order data for one period, Correlation matrix, capacity C
Result: The allocation x_i for all SKUs $i \in N$

1: For each SKU $i \in N$ that is ordered in this period, count the number of orders that contain this SKU. This number will be denoted as F_i
2: Sort the SKUs by the highest value of F_i and reset the index such that the SKU with the highest value for F_i is indexed as F_1.
3: Set $c_i = 1$ for all SKUs $i \in N$
4: **for** $i = 1$ To C **do**
5: | **for** $j = 1$ To N **do**
6: | | **if** $(j, i) \neq 0$ in correlation matrix **then**
7: | | | $c_j = c_j + \frac{|V_{O_{ij}}|}{|V_{O_i}|}$
8: | | **else**
9: | | | c_j remains unchanged
10: | | **end if**
11: | **end for**
12: **end for**
13: Calculate $a_i = F_i + \propto * c_i$ for all $i \in N$
14: Sort values from highest to lowest value of a_i and reset index such that the SKU with the highest value for a_i is indexed as a_1
15: Set $x_i = 1$ for all $i \leq C$ and

Hill-climbing local search algorithm

Next to the greedy algorithm, a hill-climbing local search algorithm is used to improve allocation. This algorithm will be run for a number of iterations (I). The results section of this thesis will explain how many iterations of this algorithm will be run. The local search algorithm is shown below:

Algorithm 2: Hill climbing local search algorithm

Input data: Historical order data for one period, capacity C, Number of iterations I, batching policy, neighborhood M
Result: The allocation for all SKUs $i \in N$

1: Set InitAlloc = Allocate SKUs based on data of a period using FoO or TOS policy
2: Set BestAlloc = InitAlloc
3: Calculate InitDistance = Travel distance per order of the period using InitAlloc, given the batching policy
4: Set BestDistance = InitDistance
5: **for** Iteration = 1 To I **do**
6: | Select two SKUs based on random location, one within M in front of the capacity border C, and one within M behind the capacity border
7: | Swap the SKUs to create CurrentAlloc
8: | Calculate CurrentDistance = Travel distance per order of the period using CurrentAlloc, given the batching policy
9: | **if** CurrentDistance <= BestDistance **then**
10: | | BestAlloc = CurrentAlloc
11: | **else**
12: | | Reverse the swap
13: | **end if**
14: **end for**

The algorithm above looks for the best possible allocation in terms of travel distance per order. With a given batching policy, these travel distances can be calculated with different allocations. Allocations will be made by swapping two SKUs. One SKU that is located in the front section is swapped with one SKU in the back section of the warehouse. Those SKUs must be located within M positions from the capacity border C. So with $C = 400$ and $M = 200$, SKUs 201-400 can be swapped with SKUs 401-600. The hill-climbing method only allows for better solutions to be saved and reverses swaps if the allocation does not perform better given the batching method. When a new allocation has been found for one period, the travel distance per order is calculated for the orders that are received in that same period to determine the performance of the allocations on order data for the same period. This is also done for the orders in the following period since in practice allocations are used for the period after which the allocation is done.

To test the performance of all the above allocation methods, an experiment will be conducted using company data from January 2020 till November 2020. This experiment allocates the SKUs to zones in the warehouse using data of one period and checks the performance of the policies for orders received in the following period.

7. Integrated design

To test the effect of the solution methods that are proposed and explained in the previous sections of the report, several experiments have been set up using a case study. In this section of the report, these experiments will be explained and the results of the experiments will be presented and interpreted. The assumptions and constraints mentioned in the conceptual model section of this report are used in the experiments in this section of the report. The different methods and algorithms discussed in Section 6 are tested using company data from January 2020 till November 2020. To be able to test this, the following values for the parameters are used:

Parameter	Value
C	400
M	200
C_{back}	76
C_{stop}	6
r_a	$2 * 50 + 5 = 165$
r_b	$2 * 80 + 5 = 165$

Figure 16: Fixed parameter values

7.1 Data analysis preparations

This section describes what data is used, how the data is cleaned, and how it is organized for modeling and deployment to find the desired results. To support this description, Figure 17 displays a flowchart that illustrates what data is used and what steps are followed to prepare and employ the data.

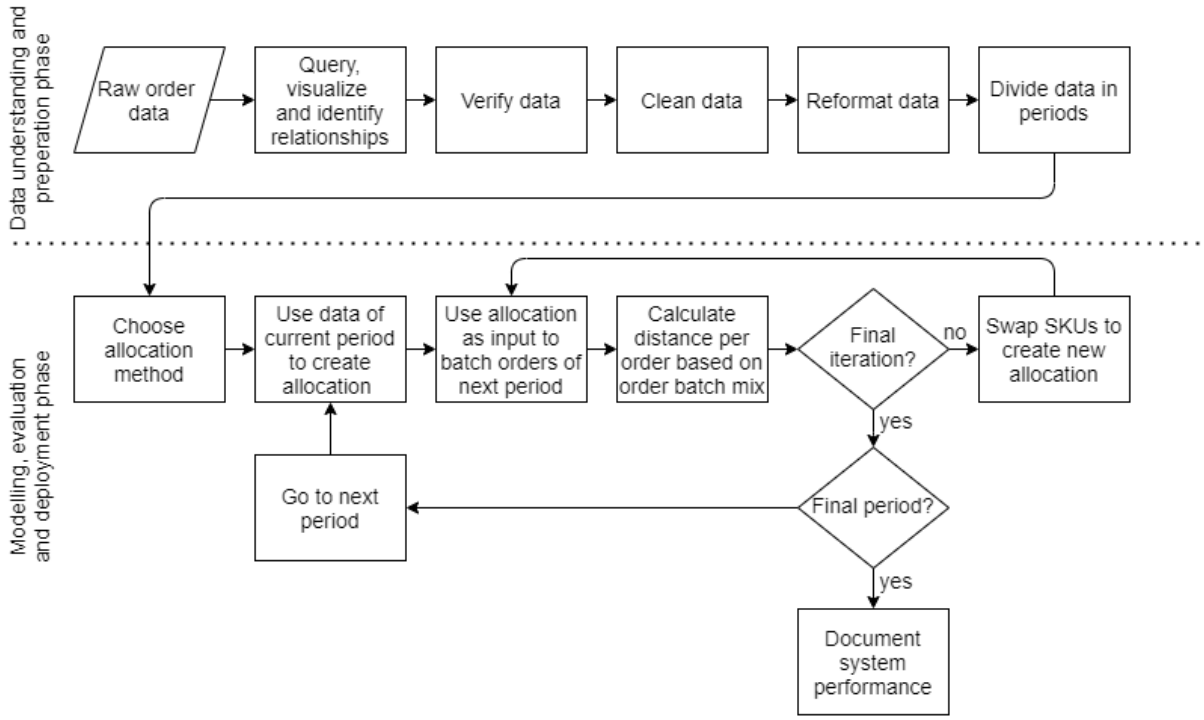

Figure 17: Data flow using CRISP-DM phases (Shearer et al., 2000)

First, raw company order data is collected. Data was available for orders that are received from January 2020 up to and including November 2020. This data is extracted out of the warehousing software used by the company. An excel worksheet with more than 400,000 rows is extracted out of the warehousing database. Each row is described as an order line; it states the article number, the article name, the order number in which this article is ordered, the ordered quantity, the date and time of ordering, and

the customer name. In Section 2.3 of this report, part of this data is queried, visualized, and used to identify relationships. The data is also verified to see how clean the data is. Two issues were found when verifying the data. Two days were identified in which no orders are received. Those days are simply skipped when analyzing the data. This will not cause any further issues. Furthermore, a few orders were found that contained more than 16 units. Since this is impossible, these orders were removed from the dataset. The remaining data is used to analyze the performance of the batching and allocation methods described in this report. Since not all information of each order line is used, the data was cleaned by removing some attributes. The attributes that are used in this research are: article number, order number, date, and quantity. To make the data easier to analyze, the order date attribute has been reformatted to day number. After reformatting, days are numbered from 1 to 335. Earlier it is explained that SKUs will be allocated after periods consisting of 28 days. The data can therefore be divided into 12 periods (11 periods of 28 days and one period of 27 days).

The remaining dataset, which is split into periods, will be used to analyze the allocation and batching methods. Data of one period is used to make SKU allocations. Data of the following period is then used to test the performance of the allocation method. The orders received in this data are batched using the given batching method. After batching, distance calculations can be executed to find the performance of both the allocation and batching method used. When using the local search method, new allocations are made for each iteration. Therefore, the batching and distance calculations also have to be executed for each iteration.

7.2 Batching method

To see the performance of the new batching method, it will be compared to the current method of batching. The allocation that is used in the current method is the TOS policy. Therefore, this comparison also uses TOS allocation. Table 3 shows the number of stops made using both batching methods for comparison and the number of AB-batches to be able to compare the two methods. Generating the batches for all periods using Python with a six-core CPU @3.2GHz takes around 15 minutes.

	Current method		**New method**	
Period	**Nr of AB batches**	**Nr of stops**	**Nr of AB batches**	**Nr of stops**
1	629	21771	457	21866
2	928	34407	689	34442
3	1671	51191	1319	51837
4	1377	44845	1096	45240
5	1095	31266	833	31542
6	795	24918	558	25175
7	561	16884	352	17065
8	501	13600	320	13854
9	579	14635	323	15017
10	706	17635	442	18093
11	933	22523	605	22899
Total	9775	293675	6994	297030

Table 3: Comparison of current batching method and the new method in terms of number of AB-batches and number of stops

To fill in the objective function given in Section 6.3, the total batches through the back section $\sum_{b \in B} z_b$ have to be found for both the current and new methods. In Table 3 we can find the number of stops per period. These numbers are summed up to find $\sum_{i \in N} \sum_{b \in B} u_{ib}$. With the given parameters C_{back} and C_{stop}, we can now fill in the objective function for both methods.

Objective function value for current method: $76 * 9775 + 6 * 293675 = 2504950$ seconds

Objective function value for new method: $76 * 6994 + 6 * 297030 = 2313724$ seconds

We can see that the new batching method is indeed an improvement over the current method since the outcome of the objective function is lower for the new method. The objective function shows that 191,226 seconds are saved over periods 1 to 11. This is a decrease of 7,6% in terms of picking time. Since the new batching method outperforms the current method, the new method will be used to calculate the main KPI: travel distance per order, for the remainder of this research.

7.3 SLAP

Allocation policies

To compare the two previously mentioned allocation policies (TOS and FoO), allocations will be made using those policies. When SKUs are allocated to their zones according to the allocation policies, incoming orders can be batched to see which of the policies performs best in terms of travel distance per order given the batching policy. Using the FoO policy, the frequency of picks in the front section of the warehouse is maximized. Therefore, the frequency of picks in the back section is minimized. However, since most orders contain multiple SKUs and orders are batched with multiple other orders, the minimization of picks in the back section does not imply that after batching, the back section is visited less using this policy. Table 4 shows the travel distance per order (MD) for both policies. The new batching method is used to find these results.

	TOS		**FoO**	
Period	$\boldsymbol{MD}$ **previous period**	$\boldsymbol{MD}$ **current period**	$\boldsymbol{MD}$ **previous period**	$\boldsymbol{MD}$ **current period**
1	23.088	24.521	23.375	24.498
2	23.314	24.785	23.824	25.095
3	23.835	25.027	24.140	25.303
4	24.303	25.256	24.563	25.484
5	23.793	24.842	24.032	25.069
6	23.263	24.056	23.384	24.341
7	22.775	23.343	22.976	23.793
8	22.516	23.567	22.354	23.525
9	21.715	22.954	21.761	22.861
10	21.057	23.651	20.845	23.321
11	21.231	23.581	21.079	23.425

Table 4: Travel distance per order for the previous and current period using the allocation of the previous period

As explained earlier, allocations are made using data of one period and used to calculate the travel distance of the following period. This table also shows the travel distance for the same period of which the data is used to allocate products. The columns that show these results are indicated by "MD previous period" in the table above. The table indicates that after batching decisions, the TOS policy outperforms the FoO policy. Both in the same period in which the allocation is made as in the following period, the travel distance per order is smaller in the majority of the periods. A t-test confirms that the mean value for the travel distance per order for the TOS policy is indeed significantly lower than the FoO policy for the previous period column. However, in practice, the allocation is made for the following period. In that case, the values of travel distance per order are not significantly better for any of the two allocation policies according to another t-test. Both t-tests can be found in Appendix D.

Greedy construction algorithm

Table 5 shows the mean travel distance per order when the greedy algorithm is used. Running the algorithm using Python with a six-core CPU @3.2GHz takes around 15 minutes of which most time is spent generating the batches. Only seconds are used to allocate products. Since the difference in using TOS or FoO policy is insignificant, the TOS policy has been used to calculate the travel distance per order using this algorithm.

Period	$\propto = 0$	$\propto = 1$	$\propto = 10$	$\propto = 30$	$\propto = 100$
1	24.521	24.521	24.521	24.551	24.551
2	24.785	24.785	24.621	24.601	24.601
3	25.027	25.050	24.999	24.993	24.993
4	25.256	25.249	25.249	25.249	25.256
5	24.842	24.842	24.669	24.513	24.522
6	24.056	24.033	23.987	23.964	23.975
7	23.343	23.343	23.343	23.343	23.343
8	23.567	23.567	23.585	23.435	23.435
9	22.954	22.954	22.969	22.969	22.969
10	23.651	23.651	23.651	23.690	23.690
11	23.581	23.562	23.581	23.581	23.581

Table 5: Travel distance per order using the greedy construction algorithm with different values of alpha

The greedy construction algorithm is used to approximate the QAP given in Section 6.4. When a higher value for alpha is used, the outcome of the objective function increases. Below, the objective function is approximated for all 11 periods that are analyzed with a $\propto$ value of 0 and 30:

$\sum_{i=1}^{N} F_i x_i + 0 * \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} P_{ij} x_i x_j = \sum_{i=1}^{N} F_i x_i = 316837$

$\sum_{i=1}^{N} F_i x_i + 30 * \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} P_{ij} x_i x_j \cong 317446.49$

However, for the actual problem, which includes batching, increasing this objective function is not the goal. A limited sensitivity analysis has been performed by using different values for alpha and see the effect of this on the travel distance per order. Table 5 shows that the travel distance per order for different values. The algorithm seems to be performing the best with $\propto = 30$. When comparing the values using the greedy algorithm with the values of only using the TOS policy (indicated in Table 5 with $\propto = 0$) only slightly improves the allocation in terms of travel distance in some periods.

Hill climbing local search algorithms

Period	TOSDistance	InitDistance	BestDistance	ActualDistance	Total orders
1	24.521	23.088	22.770	24.444	3942
2	24.785	23.314	22.938	24.690	5859
3	25.027	23.835	23.443	24.916	10542
4	25.256	24.303	23.946	24.909	8762
5	24.842	23.793	23.426	24.650	6933
6	24.056	23.263	22.856	24.050	5211
7	23.343	22.775	22.369	23.667	3702
8	23.567	22.516	21.908	23.314	3199
9	22.954	21.715	21.611	22.243	3883
10	23.651	21.057	20.466	23.575	4550
11	23.581	21.231	20.805	23.502	6214

Table 6: Travel distance per order per period using local search using TOS policy and 1,000,000 iterations

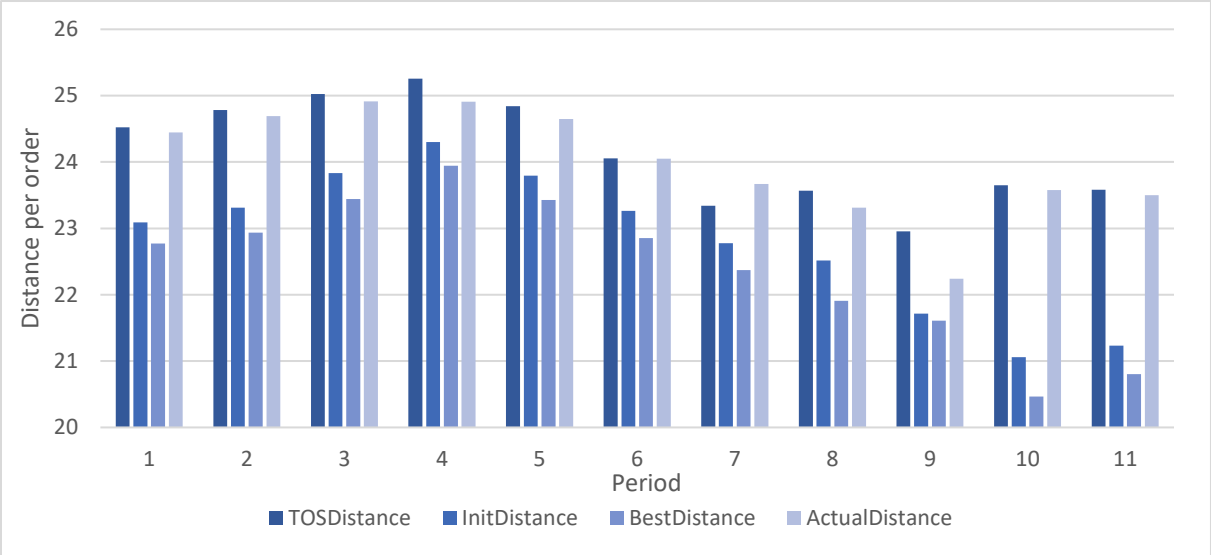

Figure 18: Travel distance per order per period using local search using TOS policy and 1,000,000 iterations

Table 6 and Figure 18 show the results of the usage of the hill-climbing local search algorithm. It shows the travel distance per order and how this distance evolves during the local search algorithm. The lower part of the data flow shown in Figure 17: Data flow using CRISP-DM phases (Shearer et al., 2000) is followed to find the results using this algorithm. The number of iterations that is used in the local search algorithm to get the results above is 1.000.000 per period. Running the algorithm using VBA with a six-core CPU @3.2GHz takes around 5 hours. TOSDistance shows the distance traveled in the current period based on turnover-based allocation using data of the previous period. InitDistance uses the allocation of the previous period and shows the traveled distance for orders in that same period. This Initial distance is improved using local search to get BestDistance. Lastly, ActualDistance shows the distance traveled per order in the current period using the allocation made by the local search algorithm in the previous period. The maximum traveled distance per order would be 27.5. This is reached when all batches in a period would have at least one SKU that is placed in the back section of the warehouse. The minimum distance would be 17.5. This is the travel distance per order if the back section of the warehouse is not visited in that period.

The table and figure confirm that the local search algorithm does find a better allocation than the TOS policy and the greedy construction algorithm do. The local search algorithm improves the greedy construction allocation by only 0.3% in terms of travel distance per order. Moreover, when looking at the ActualDistance, we see that the distance traveled in the next period based on the allocation using the local search algorithm does significantly outperform the current situation (TOSDistance). The t-test statistics in Table 16 in Appendix D show evidence of this significance. In terms of total distance traveled in all periods, ActualDistance is only an 0.6% improvement over TOSDistance. This is equal to 9,641 meters over the 11 time periods (302 days). In this research, the best performance is reached using the local search algorithm for allocation and the new batching method.

7.4 Discussion of results

In the case of Kuehne + Nagel, we see that the TOS policy slightly outperforms the FoO policy. It was expected that the FoO policy would perform better since the number of items in one order was larger than in comparable e-commerce warehouses. Typical e-commerce warehouses have 1-3 items per order. In this case, an average of 6.5 different SKUs is ordered in one order. If all items would be picked in separate picking routes, the FoO policy would outperform the TOS method. When entire orders are

picked, however, and these orders are picked in batches that contain multiple orders, the TOS policy seems to perform better.

The greedy algorithm uses the probability that SKUs are ordered together. In a situation in which the average number of different SKUs in an order is low, using this probability would be more effective. This case at K+N is special since the items do not correlate one to one. Many different combinations of SKUs in orders exist since a maximum of 16 SKUs can be ordered. Therefore using the probability of SKUs being ordered together to reallocate SKUs does not necessarily mean that there are orders consist of only those SKUs. The correlation matrix used is created using the same data that is used to find the results of the allocation methods in this report. We see that the greedy algorithm does perform slightly better than the TOS policy in the periods in which the most orders are received. This could be the case since the results of the correlation analysis are the most applicable to these periods. It is concluded that using the greedy algorithm is not optimal since using correlations, in this case, is too complex.

However, based on the few correlations that do exist in each period, local search can find allocations that actually places less frequently ordered SKUs in the front section of the warehouse. These SKUs will be placed in the front section using local search since these SKUs are, in that period, often ordered with SKUs that are ordered frequently. Local search is therefore the best performing allocation method of the methods analyzed in this research. Because of the unpredictable demand, however, the improvement that this type of allocation brings is limited. When the local search algorithm finds better allocation for one period, this does not mean that this allocation performs better in terms of travel distance in the next period. When demand is more stable, local search algorithms that operate in a similar method would become more effective.

In practice, local search algorithms can be run for a long time. In this report, 1,000,000 has been used as the number of iterations per period. Figure 19 shows that after 200,000 iterations, the results found by the algorithm do not improve in large steps any longer. Therefore, using a larger number of iterations seem to be unnecessary.

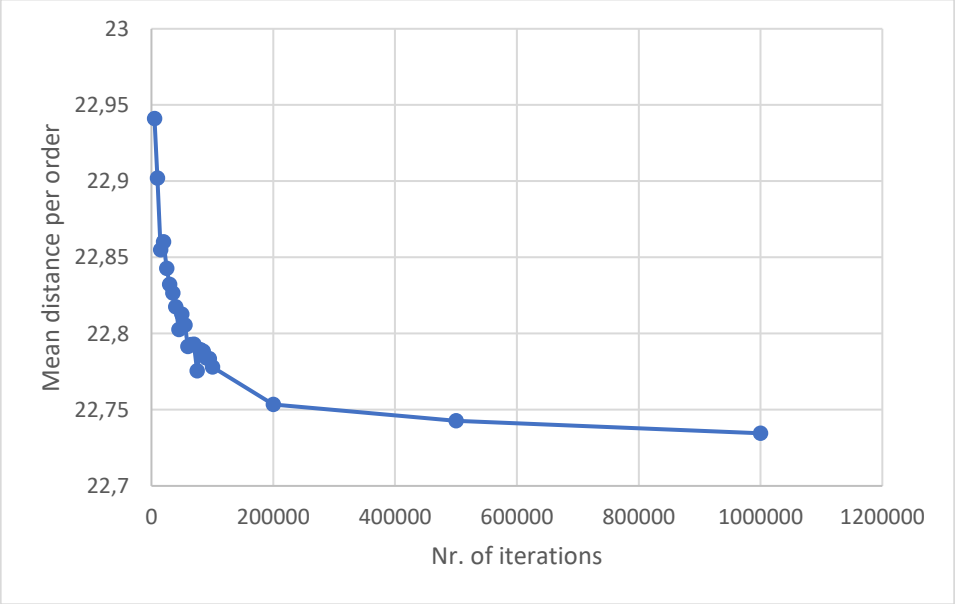

Figure 19: Mean distance per order using different numbers of iterations

Next to the number of iterations, other, more complex methods of local search can be used to find better results for the allocation. This research shows that even with simple local search methods, the

allocation for one period can be improved. However, because of the instability of the demand, the effect of the allocation improvement found by local search is limited.

The new allocation method saves 9,641 meters over the 11 time periods (302 days), which is a small saving. With a travel speed of 0.79 m/s, the time that would have been saved in this time period is 203 minutes. The new batching method however does provide significant savings. Using the new method, 2781 routes that would have traveled through the back section using the old batching method, only travel through the front section using the new method in these 11 periods. With this, 166,860 meters are saved. This is equal to 3520 minutes with the travel speed of order pickers. The number of stops did however increase using the new batching method. The extra time that is needed for those stops is only 336 minutes. In 2020, the batching method would have saved the warehouse around 65 hours of picking time.

8. Conclusion

In this research, it is investigated if the implementation of a new allocation and batching method would be beneficial for the considered company. When deciding on what allocation method should be used, it is important to see what characteristics compared to other e-commerce warehouses the company has. The newly defined Frequency of Ordering allocation policy does not seem to perform better for this company but might perform better for warehouses that receive orders that consist of a higher number of different SKUs. In the case of this company, a turnover-based policy works better than the FoO policy. Using a greedy construction algorithm that uses the probability of two SKUs being ordered together, or a hill-climbing local search algorithm does allow for a significant, but limited improvement of allocation in this situation because of the unstable demand pattern.

The local search algorithm used in this report can improve allocation for companies for which demand is stable and predictable. For this warehousing system at Kuehne + Nagel, the demand is unpredictable. Allocations made using the local search algorithm based on data of one period are therefore often not optimal for the demand that is received in the next period.

The batching method used in this research utilized the layout of this warehousing system. The current batching method only minimizes the number of stops made by pickers. The company should consider using the batching method developed in this report. The new batching method significantly improves the traveling distance. For this company, the implementation of a system that uses new allocation and batching methods would result in a decrease in traveling distance by 9.1%. In terms of hours saved, this company could save 56 hours in picking time over 302 days.

9. Limitations and future work

In this section, limitations of the current study and aspects that could be included in future research are elaborated.

- The research that is done in this report is restricted to the system explained in the conceptual design section of the report. The system consists of only two picking aisles with one cross-aisle and fixed routing. The research could be extended when a system with more picking aisles, more cross-aisles, or different routing methods can be analyzed. Also, the performance of the algorithm can be tested when the capacity of the front section of the warehouse relative to the assortment size differs. The results indicate that the FoO allocation policy could work better when a larger portion of the total assortment can be allocated in the front section. The research can also be extended to see how such a system would operate when a single location consists of multiple SKUs.
- In the research, the reallocation of SKUs is done actively. This means that when SKUs are assigned to another zone in the warehouse, they will be directly reallocated. In practice, SKUs often get allocated to a new position once the old location has been picked empty. Designing an allocation system in which the reallocation of products is done passively is more complex and would add value to literature.
- Changing the length of the period after which reallocation takes place can also add value to literature. Reallocating more or less often could be beneficial for the company. This research does not include the trade-off between reallocation costs and the travel distance savings that are made using the allocation methods.
- A limitation to the local search algorithm is that it uses one period to decide on which items are placed in which zone in the warehouse. Enlarging this optimization period could make for better performance. Since in this case, the assortment size is reduced over the year 2020 and no information about when items were no longer in the assortment is available, it could also be beneficial to make the optimization phase of the local search algorithm smaller since this will prevent products that are no longer available to be allocated to the front section of the warehouse. Also, value could be added to this research if other methods of allocation that could perform better are used as a starting solution to the local search algorithm.
- The final limitation discussed in this report is that the demand in this case study is extremely unstable and unpredictable. Conducting other case studies in which the demand of SKUs is more stable and with better information about when SKUs are available might lead to significant improvements in the performance of this system.

References

Aboelfotoh, A., Singh, M., & Suer, G. (2019). Order batching optimization for warehouses with cluster-picking. *Procedia Manufacturing*, *39*(2019), 1464–1473. https://doi.org/10.1016/j.promfg.2020.01.302

Azadnia, A. H., Taheri, S., Ghadimi, P., Mat Saman, M. Z., & Wong, K. Y. (2013). Order batching in warehouses by minimizing total tardiness: A hybrid approach of weighted association rule mining and genetic algorithms. *The Scientific World Journal*, *2013*. https://doi.org/10.1155/2013/246578

Bartholdi, J., & Hackman, S. (2011). Warehouse & distribution science 2007. *Available on Line at:/Http://Www. Tli. Gatech. Edu/ …*, *January*, 299. https://doi.org/http://www.warehouse-science.com/

Bertrand, J. W. M., Wortmann, J. C., Wijngaard, J., Suh, N. P., Jansen, M. M., Fransoo, J. C., de Kok, A. G., & de Jonge, T. M. (2016). *Design of Operations Planning and Control Systems (DOPCS)*. 192.

Boysen, N., de Koster, R., & Weidinger, F. (2019). Warehousing in the e-commerce era: A survey. In *European Journal of Operational Research* (Vol. 277, Issue 2, pp. 396–411). https://doi.org/10.1016/j.ejor.2018.08.023

Boysen, N., Stephan, K., & Weidinger, F. (2019). Manual order consolidation with put walls: the batched order bin sequencing problem. *EURO Journal on Transportation and Logistics*, *8*(2), 169–193. https://doi.org/10.1007/s13676-018-0116-0

Bozer, Y. A., & Kile, J. W. (2008). Order batching in walk-and-pick order picking systems. *International Journal of Production Research*. https://doi.org/10.1080/00207540600920850

Cano, J. A., Correa-Espinal, A. A., & Gómez-Montoya, R. A. (2017). An evaluation of picking routing policies to improve warehouse efficiency. *International Journal of Industrial Engineering and Management*, *8*(4), 229–238. www.iim.ftn.uns.ac.rs/ijiem_journal.php

Choy, K. L., Sheng, N., Lam, H. Y., Lai, I. K. W., Chow, K. H., & Ho, G. T. S. (2014). Assess the effects of different operations policies on warehousing reliability. *International Journal of Production Research*, *52*(3), 662–678. https://doi.org/10.1080/00207543.2013.827807

Chuang, Y. F., Lee, H. T., & Lai, Y. C. (2012). Item-associated cluster assignment model on storage allocation problems. *Computers and Industrial Engineering*, *63*(4), 1171–1177. https://doi.org/10.1016/j.cie.2012.06.021

Clement, J. (2019). • *Global retail e-commerce market size 2014-2023 | Statista*. Statista. https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

Cortés, P., Gómez-Montoya, R. A., Muñuzuri, J., & Correa-Espinal, A. (2017). A tabu search approach to solving the picking routing problem for large- and medium-size distribution centres considering the availability of inventory and K heterogeneous material handling equipment. *Applied Soft Computing Journal*, *53*, 61–73. https://doi.org/10.1016/j.asoc.2016.12.026

Dallari, F., Marchet, G., & Melacini, M. (2009). Design of order picking system. *International Journal of Advanced Manufacturing Technology*, *42*(1–2), 1–12. https://doi.org/10.1007/s00170-008-1571-9

de Koster, R., Le-Duc, T., & Roodbergen, K. J. (2007). Design and control of warehouse order picking: A literature review. In *European Journal of Operational Research* (Vol. 182, Issue 2). https://doi.org/10.1016/j.ejor.2006.07.009

Dijkstra, A. S., & Roodbergen, K. J. (2017). Exact route-length formulas and a storage location

assignment heuristic for picker-to-parts warehouses. *Transportation Research Part E: Logistics and Transportation Review*, *102*, 38–59. https://doi.org/10.1016/j.tre.2017.04.003

Frazelle, E. H., & Apple, J. M. (1994). Warehouse operations. In J. A. Tompkins & D. A. Harmelink (Eds.), *The Distribution Management Handbook* (pp. 22.1-22.36). McGraw-Hill.

Frazelle, E. H., & Sharp, G. P. (1989). Correlated assignment strategy can improve any order-picking operation. *Industrial Engineering*, *21*(4), 33–37. http://portal.acm.org/citation.cfm?id=72479.72482

Gademann, N., & van de Velde, S. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions (Institute of Industrial Engineers)*. https://doi.org/10.1080/07408170590516917

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences). *Computers and Intractability*.

Glock, C. H., & Grosse, E. H. (2012). Storage policies and order picking strategies in U-shaped order-picking sytems with a movable base. *International Journal of Production Research*, *50*(16), 4344–4357. https://doi.org/10.1080/00207543.2011.588621

Gómez-Montoya, R. A., Correa-Espinal, A. A., & Hernández-Vahos, J. D. (2016). Picking Routing Problem with K homogenous material handling equipment for a refrigerated warehouse. *Revista Facultad de Ingenieria*, *2016*(80), 9–20. https://doi.org/10.17533/udea.redin.n80a02

Hall, R. W. (1993). Distance approximations for routing manual pickers in a warehouse. *IIE Transactions (Institute of Industrial Engineers)*, *25*(4), 76–87. https://doi.org/10.1080/07408179308964306

Hansen, J. R., Fagerholt, K., Stålhane, M., & Rakke, J. G. (2020). An adaptive large neighborhood search heuristic for the planar storage location assignment problem: application to stowage planning for Roll-on Roll-off ships. *Journal of Heuristics*, *26*(6), 885–912. https://doi.org/10.1007/s10732-020-09451-z

Henn, S., & Wäscher, G. (2012). Tabu search heuristics for the order batching problem in manual order picking systems. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2012.05.049

Ho, Y. C., Su, T. S., & Shi, Z. Bin. (2008). Order-batching methods for an order-picking warehouse with two cross aisles. *Computers and Industrial Engineering*, *55*(2), 321–347. https://doi.org/10.1016/j.cie.2007.12.018

Hsu, C. M., Chen, K. Y., & Chen, M. C. (2005). Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Computers in Industry*, *56*(2), 169–178. https://doi.org/10.1016/j.compind.2004.06.001

Hua, W., & Zhou, C. (2008). Clusters and filling-curve-based storage assignment in a circuit board assembly kitting area. *IIE Transactions (Institute of Industrial Engineers)*, *40*(6), 569–585. https://doi.org/10.1080/07408170701503462

Jane, C. C., & Laih, Y. W. (2005). A clustering algorithm for item assignment in a synchronized zone order picking system. *European Journal of Operational Research*, *166*(2), 489–496. https://doi.org/10.1016/j.ejor.2004.01.042

Kulak, O., Sahin, Y., & Taner, M. E. (2012). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, *24*(1), 52–80. https://doi.org/10.1007/s10696-011-9101-8

Laudon, K. C., & Traver, C. G. (2016). E-commerce 2016: business. technology. society. In *Global Edition*.

Le Duc, T., & de Koster, R. (2005). *Travel Distance Estimation in Single-block ABC- Storage Strategy Warehouses* (pp. 185–200). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17020-1_10

Lee, M. K., & Elsayed, E. A. (2005). Optimization of warehouse storage capacity under a dedicated storage policy. *International Journal of Production Research*, *43*(9), 1785–1805. https://doi.org/10.1080/13528160412331326496

Lee, Moon Kyu. (1992). A storage assignment policy in a man-on-board automated storage/retrieval system. *International Journal of Production Research*, *30*(10), 2281–2292. https://doi.org/10.1080/00207549208948155

Li, J., Huang, R., & Dai, J. B. (2017). Joint optimisation of order batching and picker routing in the online retailer's warehouse in China. *International Journal of Production Research*, *55*(2), 447–461. https://doi.org/10.1080/00207543.2016.1187313

Li, Z., & Zhou, Z. (2013). An effective batching method based on the artificial bee colony algorithm for order picking. *Proceedings - International Conference on Natural Computation*. https://doi.org/10.1109/ICNC.2013.6818006

Liu, C. (2004). Optimal Storage Layout And Order Picking For Warehousing. *Operations Research*, *1*(1), 37–46.

Menéndez, B., Pardo, E. G., Alonso-Ayuso, A., Molina, E., & Duarte, A. (2017). Variable Neighborhood Search strategies for the Order Batching Problem. *Computers and Operations Research*. https://doi.org/10.1016/j.cor.2016.01.020

Pang, K. W., & Chan, H. L. (2017). Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, *55*(14), 4035–4052. https://doi.org/10.1080/00207543.2016.1244615

Park, B. C., & Lee, M. K. (2007). Closest open location rule under stochastic demand. *International Journal of Production Research*, *45*(7), 1695–1705. https://doi.org/10.1080/00207540600855007

Petersen, C. G. (1999). The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations and Production Management*, *19*(10), 1053–1064. https://doi.org/10.1108/01443579910287073

Petersen, C. G. (2009). AN EVALUATION OF ORDER PICKING POLICIES FOR MAIL ORDER COMPANIES. *Production and Operations Management*, *9*(4), 319–335. https://doi.org/10.1111/j.1937-5956.2000.tb00461.x

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

Petersen, C. G., & Schmenner, R. W. (1999). An evaluation of routing and volume-based storage policies in an order picking operation. In *Decision Sciences* (Vol. 30, Issue 2, pp. 481–501). Decision Sciences Institute. https://doi.org/10.1111/j.1540-5915.1999.tb01619.x

Pinto, A. R. F., & Nagano, M. S. (2019). An approach for the solution to order batching and sequencing in picking systems. *Production Engineering*. https://doi.org/10.1007/s11740-019-00904-4

Queyranne, M. (1986). Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*. https://doi.org/10.1016/0167-6377(86)90007-6

Ratliff, H. D., & Rosenthal, A. S. (1983). ORDER-PICKING IN A RECTANGULAR WAREHOUSE: A SOLVABLE CASE OF THE TRAVELING SALESMAN PROBLEM. *Operations Research*. https://doi.org/10.1287/opre.31.3.507

Reyes, J. J. R., Solano-Charris, E. L., & Montoya-Torres, J. R. (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, *10*(2), 199–224. https://doi.org/10.5267/j.ijiec.2018.8.001

Scholz, A., Henn, S., Stuhlmann, M., & Wäscher, G. (2016). A new mathematical programming formulation for the Single-Picker Routing Problem. *European Journal of Operational Research*, *253*(1), 68–84. https://doi.org/10.1016/j.ejor.2016.02.018

Shearer, C., Watson, H. J., Grecich, D. G., Moss, L., Adelman, S., Hammer, K., & Herdlein, S. a. (2000). The CRISP-DM model: The New Blueprint for Data Mining. *Journal of Data Warehousing*.

Theys, C., Bräysy, O., Dullaert, W., & Raa, B. (2010). Using a TSP heuristic for routing order pickers in warehouses. *European Journal of Operational Research*, *200*(3), 755–763. https://doi.org/10.1016/j.ejor.2009.01.036

Valle, C. A., Beasley, J. E., & da Cunha, A. S. (2016). Modelling and solving the joint order batching and picker routing problem in inventories. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9849 LNCS*, 81–97. https://doi.org/10.1007/978-3-319-45587-7_8

Won, J., & Olafsson, S. (2005). Joint order batching and order picking in warehouse operations. *International Journal of Production Research*, *43*(7), 1427–1442. https://doi.org/10.1080/00207540410001733896

Yaman, H., Karasan, O. E., & Kara, B. Y. (2012). Release time scheduling and hub location for next-day delivery. *Operations Research*, *60*(4), 906–917. https://doi.org/10.1287/opre.1120.1065

Yang, P., Miao, L., Xue, Z., & Ye, B. (2015). Variable neighborhood search heuristic for storage location assignment and storage/retrieval scheduling under shared storage in multi-shuttle automated storage/retrieval systems. *Transportation Research Part E: Logistics and Transportation Review*, *79*, 164–177. https://doi.org/10.1016/j.tre.2015.04.009

Yang, P., Zhao, Z., & Guo, H. (2020). Order batch picking optimization under different storage scenarios for e-commerce warehouses. *Transportation Research Part E: Logistics and Transportation Review*, *136*. https://doi.org/10.1016/j.tre.2020.101897

Yoon, C. S., & Sharp, G. P. (1996). A structured procedure for analysis and design of order pick systems. *IIE Transactions (Institute of Industrial Engineers)*. https://doi.org/10.1080/07408179608966285

Yu, M., & De Koster, R. (2010). Enhancing performance in order picking processes by dynamic storage systems. *International Journal of Production Research*, *48*(16), 4785–4806. https://doi.org/10.1080/00207540903055693

Zhang, R. Q., Wang, M., & Pan, X. (2019). New model of the storage location assignment problem considering demand correlation pattern. *Computers and Industrial Engineering*, *129*(December 2018), 210–219. https://doi.org/10.1016/j.cie.2019.01.027

Žulj, I., Kramer, S., & Schneider, M. (2018). A hybrid of adaptive large neighborhood search and tabu

search for the order-batching problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2017.06.056

Appendices

Appendix A: Batching algorithm example

The following is an example of the batching algorithm given in Section 6.3. Assume that 12 orders have arrived in a day. To simplify this problem, all orders consist of 4 different SKUs of which 4 units are ordered (i.e. each order consists of 16 units). In this example, 20 different SKUs consist.

The allocation decisions in this example state that SKU 1-15 are located in the front section ($x_i = 1$) and SKU 16-20 in the back section ($x_i = 0$) of the warehouse. The following table shows the SKUs of all orders o without the quantity of the SKUs:

o	SKU	SKU	SKU	SKU	Pool
1	2	4	12	20	AB
2	1	5	8	13	A
3	5	7	8	14	A
4	7	10	11	15	A
5	1	2	3	4	A
6	4	6	16	19	AB
7	1	2	3	4	A
8	6	8	9	11	A
9	2	3	5	6	A
10	2	3	4	6	A
11	8	11	12	15	A
12	2	6	10	18	AB

Table 7: SKUs per order

According to the batching algorithm we start with $o = 1$ and check if all SKUs are in the front section, this is not the case. Thus, order 1 will be added to AB-orders. This step is repeated for all orders. The table above shows which order pool each order is added to according to steps 1-10 of the batching algorithm.

Step 11 of the algorithm checks if there are more than six orders left in A-order pool. At the start, this is more than six so we start batching for orders {2,3,4,5,7,8,9,10,11}. Between all pairs of those orders, the Jaccard coefficients are calculated in step 12. These coefficients are shown in Table 8.

	2	3	4	5	7	8	9	10	11
2									
3	0,33								
4	0,00	0,14							
5	0,14	0,00	0,00						
7	0,14	0,00	0,00	1,00					
8	0,14	0,14	0,14	0,00	0,00				
9	0,14	0,14	0,00	0,33	0,40	0,14			
10	0,00	0,00	0,00	0,60	0,60	0,14	0,60		
11	0,14	0,14	0,33	0,00	0,00	0,14	0,00	0,00	

Table 8: Jaccard coefficient of order pairs

In step 13, the order pair with the highest Jaccard coefficient is combined. According to the above table, orders 5 and 7 are combined. The Jaccard coefficients of all orders with this batch are first calculated in step 15 and shown in Table 9. In step 16 the order with the highest coefficient with the combination is added to the batch. Order 10 will now be added to the batch. Steps 15 and 16 are repeated until 6 orders are in the batch. Table 9 shows all steps that create the first batch.

	{5,7}
2	0,14
3	0,00
4	0,00
8	0,00
9	0,33
10	0,60
11	0,00

	{5,7,10}
2	0,13
3	0,00
4	0,00
8	0,13
9	0,50
11	0,00

	{5,7,9,10}
2	0,25
3	0,11
4	0,00
8	0,11
11	0,00

	{2,5,7,9,10}
3	0,20
4	0,00
8	0,20
11	0,09

Table 9: Jaccard coefficient calculations per step

When 6 orders are in the batch. The batch (combination) will be saved and the orders will be removed from the order pool. The combination that is removed from the order pool is {2,3,5,7,9,10}. Since only 3 orders remain in the A-orders pool, step 11 of the algorithm states that we will go to step 19. All remaining A-orders are added to the AB-order pool. Since the number of remaining orders in the AB-orders pool is not more than six, step 31 will be executed. The remaining orders in this pool will therefore be batched. In this example, the two batches that are created are {2,3,5,7,9,10} and {1,4,6,8,11,12}.

Appendix B: Correlation matrix

The following table shows the correlation coefficients of the SKUs that have a coefficient. All other combinations of SKUs will have a coefficient of 0.

SKU *i*	**SKU *j***	**Coefficient**
723830000100	723830000216	0,263596276
723830000216	723830000100	0,428343949
87232172	87232363	0,516506922
87232363	87232172	0,311696658
723830000100	0723830778139	0,237138658
0723830778139	723830000100	0,48839556
5411098730314	87232363	0,283118406
87232363	5411098730314	0,310411311
5411718912236	5411718912250	0,475987193
5411718912250	5411718912236	0,469473684
723830000216	0723830778139	0,347929936
0723830778139	723830000216	0,440968718
5411718912274	5411718912236	0,380514706
5411718912236	5411718912274	0,441835646
5411718912274	5411718912250	0,375919118
5411718912250	5411718912274	0,430526316
4260473460060	7446022822894	0,327022375
7446022822894	4260473460060	0,351526364
723830000100	0723830770010	0,185693288
0723830770010	723830000100	0,330715532
87232172	5411098730314	0,389776358
5411098730314	87232172	0,214536928
5411718912311	5411718912250	0,43765586
5411718912250	5411718912311	0,369473684
5411718912298	5411718912311	0,431673052
5411718912311	5411718912298	0,421446384
723830000100	87232363	0,164135228
87232363	723830000100	0,21529563
8714799404728	8718868044006	0,147335423
8718868044006	8714799404728	0,405172414
0723830778139	0723830770010	0,326942482
0723830770010	0723830778139	0,282722513
5411718912298	5411718912236	0,412515964
5411718912236	5411718912298	0,344717182

Table 10: Correlation matrix

Appendix C: Greedy algorithm example

The following is an example of the greedy algorithm given in Section 6.2 that allocated SKUs to zones in the warehouse:

Order data is received for one period. To simplify this example problem, we assume that 20 different SKUs have been ordered in the considered period. In this example the capacity in zone A, denoted as C, is set to 12. Step 1 and 2 of the greedy algorithm makes an initial allocation based on the Frequency of Ordering or the SKUs. Table 11 shows the F_i values for each SKU.

i	SKU	F_i
1	1	200
2	2	192
3	5	183
4	4	175
5	15	164
6	6	158
7	7	151
8	16	132
9	9	129
10	17	113
11	19	112
12	8	100
13	13	87
14	14	74
15	3	50
16	11	44
17	18	38
18	20	28
19	12	26
20	10	5

Table 11: F_i for each SKU

For the SKUs with $i = 1$ to 12 in Table 11, the correlations are checked. c_j is first set to 1. If any of the SKUs correlate, the value c_j for the correlating SKU j will be set to $c_j + confidence = c_j + \frac{|V_{O_{ij}}|}{|V_{O_i}|}$. Table 12 shows the correlations between SKUs and the corresponding confidence value.

SKU i	SKU j	$Confidence$
1	4	0.58
2	6	0.51
2	9	0.42
2	14	0.58
4	1	0.48
6	2	0.21
7	13	0.65
9	2	0.42
13	7	0.24
14	2	0.29

Table 12: correlated SKUs with confidence values

When values for c_i are all calculated, $a_i = F_i + \propto c_i$ will be calculated in step 13. Step 14 sorts the values for a_i from highest to lowest and resets the index for i. These values are shown in Table 13 Finally, in step 15 we set $x_i = 1$ for all $i \leq C$. All SKUs with value $x_i = 1$ are now allocated to the front section of the warehouse. All other SKUs are allocated to zone B. In this example, SKU 1, 2, 4, 5, 6, 7, 9, 13, 14, 15, 16, and 17 are allocated to zone A, and SKU 3, 8, 10, 11, 12, 18, 19, and 20 are allocated to zone B.

i	SKU	a_i
1	1	244,4
2	2	240,9
3	4	222,4
4	6	203,3
5	9	183,2
6	5	183
7	13	171,6
8	15	164
9	7	151
10	16	132
11	14	121,4
12	17	113
13	19	112
14	8	100
15	3	50
16	11	44
17	18	38
18	20	28
19	12	26
20	10	5

Table 13: SKUs sorted on Ai value

Appendix D: Comparison of allocation policies using t-tests

This appendix shows the t-tests that compare the values of the travel distance per order of the two different allocation policies that are discussed in this thesis (TOS and FoO). Table 14 shows that the values for travel distance per order are significantly lower when using the TOS policy. The values that are tested are travel time per order for the same period of which the data is used to make the allocation. Table 15 is created using values for the travel time per order for the following period (i.e. the allocation is made using data of one period and is used to batch orders and calculate the travel distance for the next period). Table 15 shows the insignificant difference of values of the 11 periods analyzed in this report.

	FoO	***TOS***
Mean	22,93936364	22,80818182
Variance	1,602148455	1,163537164
Observations	11	11
Pearson Correlation	0,993688137	
Hypothesized Mean Difference	0	
df	10	
t Stat	1,903623362	
P(T<=t) one-tail	0,043053149	
t Critical one-tail	1,812461123	

Table 14: T-test statistics showing the significantly lower values of travel time per order for TOS policy

	FoO	***TOS***
Mean	24,24681818	24,14390909
Variance	0,829348564	0,601483891
Observations	11	11
Pearson Correlation	0,971357721	
Hypothesized Mean Difference	0	
df	10	
t Stat	1,408501988	
P(T<=t) one-tail	0,094653281	
t Critical one-tail	1,812461123	

Table 15: T-test statistics showing the insignificant difference of the values of travel time per order for both allocation policies

	LS	***TOS***
Mean	23,99636364	24,14390909
Variance	0,686059055	0,601483891
Observations	11	11
Pearson Correlation	0,953000396	
Hypothesized Mean Difference	0	
df	10	
t Stat	-1,94709488	
P(T<=t) one-tail	0,040066667	
t Critical one-tail	1,812461123	

Table 16: T-test statistics showing the significantly lower values of travel time per order when using local search allocation compared to the TOS policy