IMPROVING THE WAREHOUSE PICKING PROCESS AT FLASCHENPOST

R. Grgac

S2710099

BSc Industrial Engineering and Management

University of Twente

7.7.2024

UNIVERSITY OF TWENTE.

Colophon

FACULTY

Behavioural, Management and Social sciences

DATE

7.7.2024

VERSION

Version 2

AUTHOR

Robert Grgac, s2710099

SUPERVISORS

Stephan Meisel (first supervisor)

Breno Alves Beirigo (second supervisor)

Martin Wölck (company supervisor)

EMAIL

r.grgac@student.utwente.nl

POSTAL ADDRESS

P.O. Box217

7500 AE Enschede

WEBSITE

www.utwente.nl

FILE NAME

Grgac_BA_BMS.pdf

Acknowledgments

Dear reader,

In this report, I discuss research for my final assignment related to the Bachelor of Industrial Engineering and Management at the University of Twente. For my final project, I worked with a company called Flaschenpost to help them improve their warehousing operations.

For this thesis, I would like to give special thanks to my mentor, Stephan Meisel, who guided me through it, providing me with valuable insight and giving me directions whenever I got stuck. I would also like to thank my second supervisor, Breno Alves Beirigo, who gave me great feedback and advice on how to make my research even better.

Finally, I thank the Flaschenpost company for providing me with this case and allowing me to solve a real problem and make a meaningful contribution, as well as gaining invaluable work experience. From the company, I am also very thankful to my company supervisor, Martin Wölck, who answered all my questions, of which there were many, and who was always willing to lend a hand and years of experience to help me navigate through the company processes and the problem that I was presented with.

Robert Grgac

July, 2024

Management summary

This research is a collaborative effort with Flaschenpost, Germany's largest online grocery retailer. We value the company's expertise and are working together to enhance its operations. Flaschenpost, with its warehouses spread across Germany, specialises in last-minute delivery of groceries to customers' houses. Its renowned business model promises the delivery of groceries within two hours of placing the order.

Problem

Due to the company's business model and substantial growth, it needs help with the number of orders they are processing daily. Around 20% of potential sales in orders get lost due to capacity constraints. After a detailed analysis of the company operations, the capacity bottleneck was found in the warehousing department, specifically in the warehouse picking process. From there, a more in-depth insight was gained into the warehouse picking process and potential problems that could cause a low warehouse capacity, causing the missing of potential orders. From this, the following research question was formed:

How to reduce the average order picking time in the warehouse picking process of Flaschenpost?

Solution

Our research, based on a comprehensive literature review, proposes the development of new batching and routing algorithms as the most effective solution to enhance the warehouse picking process. By optimizing the routing algorithm, we significantly reduced the pickers' walking distance, thereby accelerating the order fulfilment process. The batching algorithm, on the other hand, enables the simultaneous collection of multiple orders, further expediting the order fulfilment process. These improvements directly translate to increased warehouse capacity and a reduced ratio of missed sales opportunities.

To implement these solutions and to increase the warehouse capacity, this research provides three deliverables:

- Time-savings heuristic for batching
- Priority policy for collection of orders
- Dynamic programming based routing algorithm

A time-saving heuristic provides a local optimum approach for selecting the batches that save the most time. This approach, combined with the priority policy created, provides a sequence in which the orders should be collected and minimizes order delay. Once the batches have been created and sorted, when the warehouse worker is available, they are given the next batch in line, for which a route is constructed and provided to the warehouse picker.

Results

To measure the performance of the solutions generated in this research, we have benchmarked their performance against the current company batching and routing algorithms. To be able to measure, a simulation environment was created, which was inspired by the warehouse layout of the company, where several variables could be changed, including the number of aisles, the number of products in a warehouse, and the number of orders. By changing the parameters of the warehouse, we could see how different combinations of batching and routing algorithms perform. This analysis concluded that the best-performing combination was the time-savings batching algorithm combined with the dynamic programming routing algorithm, which resulted in a 67.5% reduction in route lengths and the

time it takes to collect each batch when compared against the benchmark company algorithms.

Conclusion

The resulting solutions proposed in this research should provide the company with a faster and more efficient warehouse picking process. By reducing the time, it takes to collect and fulfil orders to a third of what it used to be, the warehouse capacity should increase by three times if the benchmark is to be used as the current representation of the state of the company. This will result in a reduction of lost potential sales and a reduction of labour costs since fewer people will be needed to fulfil the orders, thus resulting in higher revenue and lower operations costs.

Table of Contents

Table of Figures

List of Abbreviations

FMCG – Fast Moving Consumer Goods

IEM - Industrial Engineering and Management

MPSM – Managerial Problem-Solving Method

TSP – Traveling Salesman Problem

DP – Dynamic Programme

LP – Linear Programme

NP – Nondeterministic Polynomial Time

FIFO -First In, First Out

BPM – Business Process Model

KPI – Key Performance Indicator

1 Introduction

In this first chapter, we introduce the company Flaschenpost, which has presented us with the problem we will solve in this thesis. Here, we will become familiar with the context of the problem and the critical decisions that have been made to help us structure the research for this thesis. In section 1.1, we provide some background information about the company and are introduced to the problem that the company faces in section 1.2. Once we have defined a problem, we discuss what methodology we plan to use to solve this problem in section 1.3. Sections 1.4 and 1.5 provide us with the main research question and sub-questions, which will help us break down the significant problem presented by the company into a set of more minor, efficiently addressable problems. After we have divided the problem into smaller ones, we look at the company's current state and get a deeper insight into its operations in section 1.6. In sections 1.8 and 1.9, we discuss the KPIs that we will use to measure the company's performance and how we ensure that the data we are measuring is valid and reliable. Finally, we discuss the conclusions of this chapter and how we plan to use them in the remainder of our thesis.

1.1 About Flaschenpost

The problem we were presented with was from Flaschenpost[1]. Flaschenpost is a German company that was founded in 2016. The company initially started as a beverage supplier, and due to its massive success, it has turned into an online supermarket as well as the largest Fast Moving Consumer Goods (FMCG) supplier in Germany in 2022 (ecommerceDB.com, 2023). Because of the fast growth rate, the company was bought by the Oetker Group in 2020, and to this day, the company keeps working and collaborating with other businesses and brands.

Today, the company delivers groceries to people in over 190 cities across Germany. It owns over 30 warehouses and plans to expand more in the future, even outside of Germany. It currently has around 20,000 employees and fulfils over 10 million orders per year. But what really sets it apart is its business model, which promises to deliver orders from the website to the customer's door within 2 hours.

While the company's achievements are impressive, from the sheer scale of its operations with many different warehouses to its ambitious business model, many logistical issues need to be overcome for it to fulfil its promises. To help them in this venture, the company has allowed me to provide specific solutions for their warehousing operations. More specifically, how can these warehouse pricking processes be modified to improve their performance?

1.1.1 Context

Before we get into the details of the company's problem, it is important that we understand why this problem is relevant in the industry today and how even small improvements can save a lot of resources.

Just last year, the FMCG market accounted for 7.5% of Germany's total food market revenue (GfK, 2024). When we combine this with the information that in Germany in 2023, the total food revenue was around 197.6 billion euros (GfK, 2024), from this, we get that the yearly

[1] The infromation about the company came from their website: https://www.flaschenpost.de/unternehmen/ueber-uns

revenue of the FMCG market just is currently estimated to be around 14.82 billion euros. Aside from these large numbers, we can also see that in the last five years, the number of people who have been purchasing their groceries online has been steadily increasing; throughout this period, just in Germany, the number of people who regularly order groceries has increased by 1.7 million (IfD Allensbach, 2023).

From all this data, it can be concluded that the companies' sales and demand have been rapidly increasing. Because of that, certain parts of the company's infrastructure need to be changed to cope with this newly created demand. Throughout this thesis, my goal is to help this company achieve its target and gain more insight into warehousing processes.

1.2 Problem Description

The company handles millions of orders, all needing to be sorted and processed. Since the company has over thirty warehouses, before the customer even starts to order, they are first sorted into a region that belongs to the warehouse closest to them. This information is vital because different warehouses have different capacities at a given time. Before the customer can place an order, the warehouse needs to check if it has enough capacity to handle new demand; if that is possible, the customer then places an order, which is then sent to the warehouse to be processed. Once the order reaches the warehouse, it is placed on a list and waits to be collected by a picker. Once the order is collected by a picker and delivered to the depot, it is loaded into a truck, and the remaining orders are waiting to be loaded before the truck departs. A truck follows the previously constructed route and delivers the orders to the customer's door.

With this overview of who the company is and how its orders are being processed, it will be easier to understand its problems. Due to the significant increase in demand, the infrastructure the company has used so far has reached the limits of its capacity. Over the past few years, the focus has been optimizing vehicle routing, the last phase of the operations before the customer receives their order. However, a high focus on vehicle routing optimization left the optimization of other processes in the warehousing operations neglected, thus leading to certain bottlenecks.

To fix this, the company has decided to start a new project on warehousing operations with the final goal of increasing its warehouse capacity. They would like to find out what approaches and methods exist that could be implemented to reduce time and workforce, or ideally, both, when it comes to processing orders.

1.2.1 Norm vs. Reality

According to Heerkens (2017), when defining a problem, it is important to know the difference between the current situation (the reality) and the desired state (the norm). To put this into the perspective of our company and thesis, the missed order ratio of the company is less than the company desires. We define the order missed order ratio as one minus the ratio of the placed and fulfilled orders out of all the potential orders that the company can get.

The main reason for the loss of potential orders is the lack of capacity. As described in the problem description, before the customer can buy the groceries, the system checks if it has enough vehicles, stock, and workforce in the warehouse to deliver the order placed within two hours. If that is not the case, the system will notify the customer and provide them with an option on the website to have their groceries delivered after the two-hour window, which means that the system will register the order and place it automatically for the customer when there is more capacity. However, some customers are not willing to wait longer than two hours and decide to cancel their order completely. This phenomenon is called shopping cart

abandonment (Kapoor & Vij, 2021). While shopping cart abandonment can be caused by a wide variety of reasons, such as a complicated checkout process or lack of a user-friendly interface, after several discussions with the company, the main reason for the card abandonment, in this case, is the fact that customers must wait to place their orders. That is why the company wants to decrease the missed orders ratio to as much as possible to reduce the number of lost potential sales thus achieving their norm.

1.2.2 Problem Cluster

As described in the previous section, the company focuses on reducing the missed orders ratio as much as possible. To help us map out the cause and effect of the company's problem, we have decided to make this our action problem. According to Heerkens (2017), the action can be defined as a gap between the norm and the reality, as seen from the perspective of the problem owner. To help us find the origin of this problem, we will make a problem cluster to narrow down the exact cause of this problem, as can be seen in *Figure 1.*

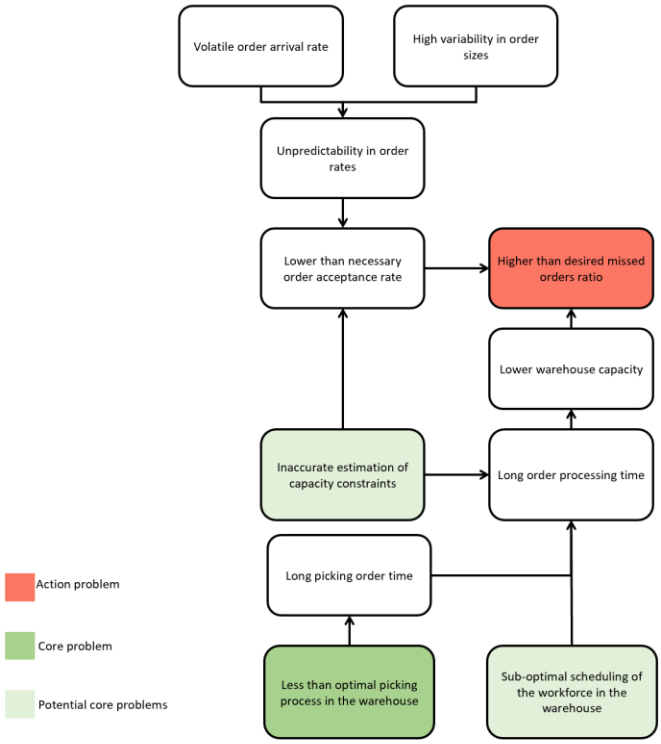

Figure 1: Problem Cluster

From this problem cluster, we can find several reasons for the high missed orders ratio. Firstly, the volatile order rate and high variability in order sizes make it difficult to know how many orders can be accepted at any given time. Because of that, sometimes more or fewer orders are accepted than are strictly necessary. Secondly, the order processing time is longer than it needs to be. When we talk here about the order processing time, we refer to the time it takes to process an order in the warehouse, from when the order arrives to when it is placed in the van and sent to be delivered. We stop at the vehicle routing because the company has extensively optimized this process.

If we investigate the causality of these issues a layer deeper, several potential core problems begin to emerge. A core problem, as defined by Heerkens (2017), is the main problem we will solve. While the action problem is the problem owner's main issue, the problem's real cause

might be something else. In a problem cluster, there can be multiple core problems; however, for this thesis, we will only select one.

The first potential core problem is the inaccurate estimation of the capacity constraints. This is a problem since the company only sometimes ideally estimates how many orders they can handle, which sometimes leads to them underestimating or overestimating their capacity. However, this does not improve the missed orders ratio by as much as we would want because the main reason for the potential lost sales is low capacity. Knowing exactly how much we can handle would allow us more “breathing room” and not take more orders than we can handle, but the capacity of the warehouse would not be changed, and the rate at which the orders arrive is not something we can control, thus making the effort/reward ratio not worth it.

The second potential core problem would be the sub-optimal scheduling of the workforce. What this means is that sometimes, in the warehouse itself, there need to be more workers to fulfil the orders, and then there are other times when the warehouse has too many workers, and their idle time is seen as a loss from the company. The goal here would be to have an optimal number of workers at all times in the warehouse such that the demand is still being met and there is minimal idle time where no work gets done. However, this process in the company is already being improved, and models for such a problem are already being made. This is good for us because, since the company has already decided to address this issue themselves, that means that they are aware of it and that we can focus our efforts on a different problem.

Finally, we get to the last potential core problem, which is the less-than-optimal picking process in the warehouse. What we here mean by the picking process is that once an order arrives at the warehouse, it is sorted and placed in a queue. Once the order progresses to the front of the queue, the worker takes the order and picks up all the items from the warehouse that belong to that order. The time it takes for the picker to collect all of the items and deliver them to a van, plus the time the order was waiting in the queue, is how we measure order processing time. This order processing time affects how long it takes for an order to be fulfilled, and the more time it takes to complete an order, the fewer orders we can fulfil in a day. Therefore, the time it takes to fulfil the orders directly affects the warehouse's capacity, and due to the shopping cart abandonment effect it indirectly affects the missed orders ratio. Because of such a significant impact on warehousing operations and its apparent connection to the extended order processing time, we have decided to take this as our core problem for this thesis. Throughout the rest of this chapter, we will explore the approach we will take, what questions we must answer, and the possible solutions we should consider.

1.3 Problem Solving Approach

In this thesis, to solve the problem that we saw in the previous section, we will use the Managerial Problem-Solving Method (MPSM). As described in the book by Heerkens (2017), this method consists of 7 phases. The MPSM method is a standard methodology used in Industrial Engineering and Management due to its versatility and ability to break down the problem into smaller and more manageable parts. In this section of the thesis, we will discuss each phase based on the work of Heerkens (2017) and how they will help us solve our problem.

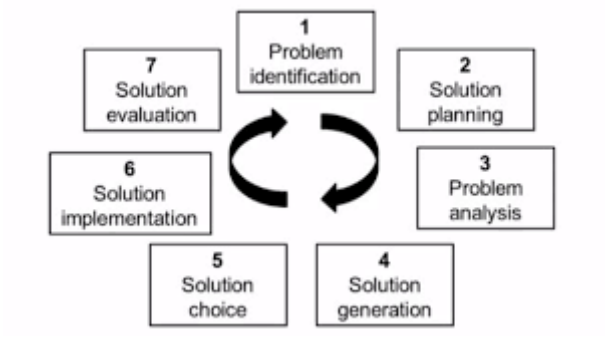

Figure 2: MPSM (Heerkens, 2017)

Phase 1: Defining the Problem– in the initial phase of the thesis, our goal is to define the specific problem that the company has. This phase of the thesis has already been done in the previous sections of this report. Often, when working with companies, the problems that the company thinks they have differ from the ones they have. That is why, in the problem description section of this report, we have developed a problem cluster to help us find the action and the core problem and help us bridge the gap between the norm and reality.

Phase 2: Formulating the Problem-Solving Approach—The second phase of the MPSM is focused on formulating the questions that will help us gain more knowledge and understanding of our problem and outline the approach we must take to reach our solution. This step will be done in the following sections of this chapter, where we will formulate our research and knowledge questions as well as certain KPIs that we would like to measure to track the improvement of our solution.

Phase 3: Analysing the Problem – The third phase of this cycle is used to re-evaluate the problem we have defined in the first phase and start doing more in-depth systematic research about the problem. In the first 2 phases, we usually are led by our intuition and what the company has told us the problem is from their perspective. However, in this phase, we aim to take a more scientific approach to this problem and start using the research cycle to help us answer some of the knowledge questions we have formed in the previous phase. These answers will help us understand the causal relationship between the KPIs and what methods of improvement currently exist.

Phase 4: Formulating Solutions– once we have gotten some further insight into what kind of solutions already exist and the causal relationships between different variables, our goal is to start forming our list of potential solutions to the problem. These solutions can be completely new or extensions of the current solution approaches, with the latter being the more common. Apart from just formulating the solutions, we also need to have a list of criteria that will help us in the later phases to determine what kind of solution would fit the most.

Phase 5: Choosing a Solution—This phase is one of the most straightforward ones. Since in the previous phase, we formulated a list of solutions and the criteria that should be used to evaluate them, in this phase, the goal is to select one of the solutions based on those criteria. It is also important to justify why that specific solution has been chosen over others.

Phase 6: Implementing the Solution—Once we have selected the solution, it is time to implement it. In this phase, we will create an implementation plan and outline the steps we should take to implement our solution. Once that has been completed, we will also elaborate on how exactly we implemented it in practice since sometimes those two can differ.

Phase 7: Evaluating the Solution – In the final phase of the MPSM, the focus is on evaluating the solution we implemented in the previous phase. Regarding the evaluation, we need to

focus on all six previous phases of the MPSM. This can then be further narrowed down to three key takeaways: How well was the problem defined? How good are the solutions that were provided? How well is the chosen solution implemented? Depending on the answers to these questions, we can determine whether the solution that has been chosen improves the performance based on the KPIs we chose and whether it bridges the gap between the norm and the reality that the company wants.

1.4 Main Research Question

In the problem identification section, we found our core problem by analysing the problem cluster. From there, we found that the main issue was the warehouse picking process. Based on these observations, we formed the following research question.

How to reduce average order picking time in the warehouse picking process of Flaschenpost?

This research question will focus specifically on the process of item collection in the warehouse. The main goal will be to reduce the time it takes to collect the items. By reducing the time, we increase the capacity of the warehouse and, therefore, reduce the number of lost potential sales, thus aligning us with the company goals and its action problem.

1.5 Sub-research Questions

We will now go in detail through each of the knowledge questions and explain what their role is in our thesis. Since the research question we created in the previous section is quite extensive, it will take time to answer it. That is why we have created a list of knowledge problems, which can be perceived as a list of research sub-questions that will help us answer our main research question. The idea behind this research question is to divide this thesis's work into three main parts. The first part is getting familiar with the problem; the second part is trying to find existing solutions and create our own; the third and final part will focus on our solutions' quality. We will thoroughly review each knowledge question and explain their role in our thesis.

1. *What are the current methods and models being used in the company for the picking process?*

The first question is supposed to help us understand how the company currently does its picking process. This question is closely tied with phase three of the MPSM, where we are trying to gain in-depth knowledge of the problem and hopefully uncover any other potential constraints that we might have missed at first glance. From the answer to this question, we will obtain some initial data and what parts of the process need improvement.

2. *What models and heuristics currently exist that deal with the warehouse picking problem?*

For this research sub-question, the focus is somewhere between phase three and phase four of the MPSM. This question addresses the third phase by first looking at all the possible solution approaches one could use to improve a warehouse. By doing this research, we get more in-depth knowledge of the problem and what things we should consider. The fourth phase is addressed because once we know all the possible ways to improve the warehouse picking process, we will generate a list of possible solutions that could be used in this thesis.

3. *Which KPIs are relevant for the improvement of the Flaschenpost' s order picking efficiency?*

Once we have an in-depth understanding of the problem and a list of potential solutions, the next part of our thesis will decide which solution to pick, which nicely correlates with phase five, the solution choice. To help us select the best solution for the company, we must first identify how the company evaluates its performance. To do that, we will create a list of KPIs based on the numerical and real-world performance of the mathematical models and heuristics that we have found.

4. *How can these algorithms that exists be modified such that they are applicable to our picking process?*

After implementing the solution, we created from the previous sub-question and combining that with the KPIs we selected in the third sub-question alongside some additional metrics, it is time to step into the final phase of the MPSM. The goal of phase seven is to evaluate the solution we created and show how it addresses the core problem. Doing so provides additional validity to our thesis and the chosen approaches.

5. *How to evaluate the mathematical models and heuristics and its impact on the Flaschenpost' s warehouse picking process?*

After implementing the solution that we have created from the previous sub-question and combining that with the KPI's that we have selected in the third sub-question alongside some additional metrics, it is time to step into the final phase of the MPSM. The goal of the phase seven is to evaluate the solution that we have created and to show how it addresses the core problem. By doing so we provide additional validity to our thesis and the approaches that were chosen.

6. *What are the next steps the company should take and future recommendations on how to further improvement of our solution?*

Finally, when the entire MPSM cycle has been completed and the solutions are implemented, we will provide the company with a list of the following steps that can be taken and future recommendations. We provide this information for two reasons. Firstly, due to the scope and limitations of the thesis, not all aspects of the problem can be addressed and implemented, and secondly, in IEM, process optimization is never truly complete, and there are always more steps that can be taken to improve further.

1.6 Current State at the Company

Before we try to fix the company's problem, we must first get a firm grip on the methodology and the operations process used, specifically in the warehousing operations, since this is the aspect of the company that we are focusing on. This is helpful for two reasons: the first is to understand the constraints and limits, and the second is to pinpoint which parts of the picking process need to be improved to model these problems mathematically in later chapters and form appropriate algorithms. The following BPM (Business Process Model) has been created to understand how the company works.

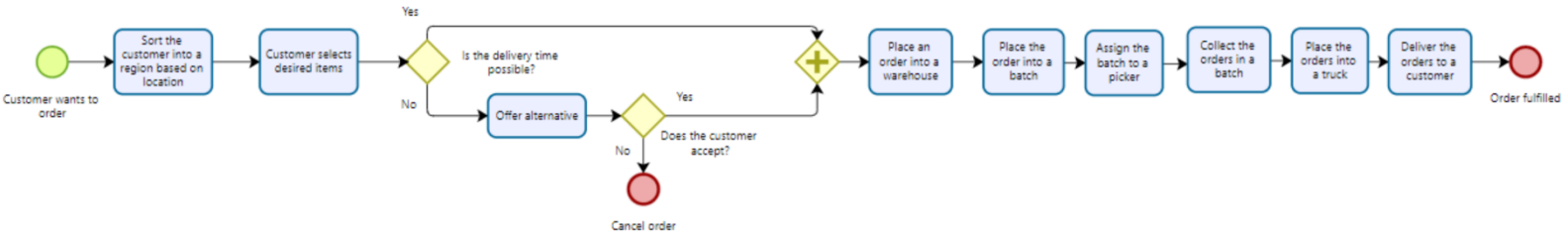

Figure 3: BPM of warehousing operations

As shown in *Figure 3,* there are two possibilities when the customer wants to place an order through the company's website. One is to place an order right now, and the other is to place an order at some point in the future. When the customer places an order, the system evaluates if it can be fulfilled at this moment or, if option two is used, if the order can be fulfilled later. The system estimates this right now by taking several factors into account. Is it possible to fulfil this order within two hours? What is the current warehouse utilisation, how many orders are in the queue right now, can the order be batched with any other order, and is there enough space in the vehicles? If the customer cannot place the order right now, an alternative option will be presented to them.

Once the customer has placed the order, it is processed by the appropriate warehouse based on the delivery address. Once the order reaches the warehouse, it will be divided into smaller components. For this, we can use the analogy of a shopping cart; when a customer places an order, it is treated as a list of groceries that a customer wants to buy. The order is treated as a "shopping cart" full of groceries; however, when packaging and delivering these groceries, they need to be placed outside of the shopping cart into grocery bags. In this thesis, we refer to these "grocery bags" as boxes. When an order arrives at a warehouse, the groceries are divided into boxes with actual grocery bags where the items are placed. These boxes are then placed into a queue to be collected. Each box will contain only the items that belong to one specific order. That means that the picker can collect boxes that belong to different orders but the contents of each box belong to one order only. These boxes are then batched together and placed in a queue waiting to be collected. When the workers pick up orders, they carry them in a cart with a maximum capacity of 6 boxes and are given a sequence in which they should pick up the items for each box. The worker then places the items in their respective boxes based on which item belongs to which order. Once the route has been completed, the worker drops off the collected items in the drop-off area, where the van drivers collect the boxes that belong to their driving route. When all the orders that belong to that van have been fulfilled and placed in said van, the groceries will be delivered to each of the respective customers.

1.6.1 Batching and Routing

Within the warehouse operation, we notice two distinct processes that take place and require substantial computational resources: batching and routing algorithms. These algorithms are crucial for warehouse operations as they consolidate orders and create routes the pickers need to follow. Both algorithms use simple heuristics that are not optimised for speed or efficiency.

The batching algorithm applies the FIFO (First in, First Out) policy. This algorithm first breaks down every order into boxes that need to be collected; sometimes, the entire order fits into

one box and other times, multiple boxes are needed. Once all the orders are split into boxes, the algorithm will find all the boxes from the oldest order and add them to the batch. After this, it will find the closest boxes within a distance of the already selected boxes and add them to the batch. The algorithm will repeat this process until there are six boxes in a batch since that is the maximum capacity of the cart.

Once the orders have been divided into boxes and consolidated into batches, a route is constructed, which provides the order in which the picker must collect the items. The routing algorithm will take one of the boxes from the batch and construct the route for the first box. Once all the items from the first box have been collected, it will then take the second box and construct a route for the second box; this process will repeat itself until all the boxes have been filled out with their respective items.

The route is formed for each box using a greedy[2] algorithm. The algorithm picks an item closest to the picker at that moment, and from there, it utilises the nearest neighbour heuristic until every item in that box is collected, and then it moves on to the next box.

1.7 KPI Selection

To evaluate the performance of our solution, we will use the company's current method and benchmark its performance against the performance of our solution. As a metric for comparison, we have listed several KPIs that could be used to track the performance of each solution. The KPIs chosen are as follows:

1. Average time it takes for one route to be completed (related to time vs distance)
2. The average distance per route (again, time vs distance)
3. Order throughput rate (how many more orders were handled per time unit)
4. Average backorder rate (relate that to how many orders out of x get backlogged)
5. Total time spent in order fulfilment (to find out what is the actual capacity of the warehouse)
6. Order delay time

The first two KPIs measure average time and distance; while they correlate, they are still not one-to-one, and both give us different information. The difference in performance measurement between time and distance is explained in more detail in the following section. The order throughput rate tells us how many orders were processed per unit of time. At the same time, this KPI would be helpful to measure since it is directly correlated to our action problem; since this company data is not accessible to us, it will not be possible to benchmark the company's performance against our data. The average backorder rate will tell us how many orders out of x get backlogged due to capacity constraints; unfortunately, it is impossible to measure this data since the company cannot provide us with it. This KPI is something that the company could measure once they implement our solution into their operations and develop the proper technical background to track it. The total time spent in order fulfilment will tell us the warehouse's capacity and how many orders the warehouse can handle in a day. The order delay time will give us further information if the warehouse is over-utilized since then we can track how much impact order backlogs have. We lack the data to track both of these KPIs and do not have any data collection method. Since four out of six KPIs cannot be tracked, valid questions could be raised about these measurements' validity and reliability. The reasoning behind why the data cannot be collected and how we will ensure that the measurements are still reliable are explained in the next section.

[2] A greedy algorithm is an algorithm that implements a greedy strategy. In a greedy strategy one makes a choice that is best at the moment (Jungnickel, 2013).

1.8 Validity and Reliability

To solve the problems we have discovered in this chapter, specific data is helpful to make our solutions as accurate as possible and measure the performance clearly. Some of this information, such as the layout of the warehouse, how items are stored in it, and the algorithms currently used in the warehouse operations, are available to us. However, there are parts of the data that we cannot access due to confidentiality or technical reasons; this data includes the history of orders, the location of each item in the warehouse and the exact layouts of the warehouses. Because we need this data, there are reasonable concerns about the validity and reliability of this research.

That is why when it comes to data (especially when measuring performance), it is essential to know that the data that will be generated to make and evaluate the solution and the methods used are based on real-life examples of data and extensive interviews with the company supervisor. According to Cooper (2014), we can look at three types of validity. The first is called content validity, and it explains to what extent our data covers the problem we have defined. The second one is called construct validity, and it explains whether the criteria we measure track the performance of the warehouse well. The third one is called criterion-related validity, and it tells us if the measurements we take accurately represent the current state of the performance.

When it comes to the solution of this thesis, which will be a mathematical model, it is essential to know that every model has limitations that separate it from reality. Throughout this thesis, the goal is to reduce these assumptions as much as possible since they reduce the quality of the construct validity, which impacts the overall quality of the results. Nevertheless, some assumptions will already have to be made. The central assumption made here is that the workers will follow the instructions and not deviate from them; secondly, while the data on the distance and the layout of the items in the warehouse can be somewhat accurately represented from the architectural plans, the walking speed, and the time it takes to pick up items will have to be approximated. These approximations, alongside other relevant data necessary for the modelling, will be based on historical or real-life data examples so that the content validity is not significantly infringed.

Finally, when it comes to the model simulation, the data we have chosen is based on past orders. We will then take both the company algorithms and the ones developed in this thesis and simulate several scenarios to ensure a reliable and accurate analysis of the performance of the algorithms. The challenge that is presented here is measuring the performance of these models. Koster, Le-Duc & Roodbergen (2006) talk about the estimation of models through time and distance. In this paper, they explain that the correlation between distance and time it is not one-to-one. The main benefit of distance is that it is more precise out of the two since the distance can be easily measured; however, time, while intuitively lower when the distance is shorter, the percentage of reduced time varies on other factors as well such as walking speed, aisle congestion and the experience of the workers. If the distance is reduced by 5% on average, the picking time might only be reduced by 3% due to varying walking speeds. This is where criterion-related validity is affected, and it is essential to use both measures when evaluating models as well as proper statistical methods to account for such variations

1.9 Conclusion

Throughout this chapter, we have introduced the company and the problems that we are facing. We have then broken down this problem into a series of subproblems, which we called research sub-questions; by answering these questions in the following chapters, we will slowly build up to the solution we need. Aside from breaking down the problem, we have also gained detailed insight into the company operations and algorithms they used, which have helped us narrow down the exact mathematical problems in the company. We will now use this information to form a theoretical framework and have a more structured approach to our solution.

2 Literature Review

Once we have identified the problem, we have decided on our methodology and designed the research. This chapter will discuss the theoretical framework we have formulated for this thesis and provide a clear overview of the literature relevant to the warehouse picking problem. This chapter will consist of four sections; the first three sections divide the theoretical framework into three main sections. These sections have been derived from the literature overview of Koster, Le-Duc & Roodbergen (2006). Throughout this chapter, we will base the theoretical framework on the works of this paper. With that in mind, the warehouse picking problem can be looked at from three aspects: the warehouse design, batching and zoning, and routing. Each of these sections is quite detailed and complex; however, the point of this chapter is to provide a brief overview of each of them and show different approaches to the warehouse picking problem and the specific problems these three different frameworks solve. The final section of this chapter will be the conclusions, which explain which aspect of this framework will be used and why those have been selected for this thesis.

2.1 Warehouse Design

The first aspect through which we can look at the problem focuses on the design of the warehouse. When we talk about warehouse design, we refer to two parts. The first part focuses on the layout itself; what we mean by that is the number of aisles that the warehouse has, how they are placed around the warehouse, whether the warehouse is automated (also known as parts-to-picker systems) or run by humans (picker-to-parts systems). If humans run it, do they need to use vehicles to move around the warehouse, or is it small enough to walk just around? Can humans carry the items, or do they need a machine? Such decisions are crucial for travel time and distance that will have to be passed. This problem is also known as the "internal layout design configuration problem" (Koster, Le-Duc, & Roodbergen, 2006), and many models exist that try to tackle this issue.

The second part of the warehouse design focuses on the storage assignment policies. These policies are used to decide how we want to place items around the warehouse; these decisions are essential for efficient operations. While many policies exist, we will list five main ones according to (Islam & Uddin, 2023) and briefly discuss how they work. *Random storage* means that items are placed randomly throughout the warehouse. *Closest open location storage*: This method is often used in small warehouses; it takes the closest available spot from the entrance and places the item there. *Dedicated storage* is used when every item has a specific place in the warehouse and does not change over time. *Full turnover storage* means that the items close to the entrance are the ones that are ordered most often; such storage policy is quite dynamic and needs to be regularly updated. *Class-based storage* policy utilizes the fact that some items are often ordered together and are, therefore, placed close to each other. Each of these policies has specific pros and cons and depending on the routing method used and the warehouse layout, different ones are chosen. While there is much research on this topic, more is needed to reduce the company's order processing time significantly. Nevertheless, a policy will be provided in Chapter Six of the thesis.

2.2 Batching and Zoning

The second approach we could tackle when trying to improve warehousing operations is pre-routing preparation. This means that, before the construction of the shortest route begins, several decisions can be made to reduce the total workload of the warehouse employees.

Three approaches can be considered when it comes to pre-routing: zoning, batching, and a combination of both.

Zoning means that the warehouse is divided into several zones based on the layout of the warehouse. One or multiple workers are assigned to each zone; these workers then only traverse the area of their zone and collect items from that zone. This way, the order is split into smaller sub-orders based on the warehouse zones, where each worker only fulfils a part of the more significant order. The pros of such an approach are less travelling time for the warehouse workers and faster collection of items; however, the downsides are that when the order is split into smaller parts, the synchronization of different orders and matching all the sub-orders creates additional work. Some zones might be ordered from more often than others, which could lead to an uneven workload distribution.

The second option, batching, is used when multiple orders arrive at the warehouse. In these cases, it is sometimes possible to combine some of the orders such that instead of the worker going twice to the same place in the warehouse, they make a longer route and collect several orders simultaneously. This approach saves both time and resources, and according to Koster, Le-Duc, and Roodbergen (2006), any kind of batching using even some simple heuristics is better than no batching. There are two main approaches when it comes to batching: one is making a linear programme (LP) model, and the other one uses heuristic algorithms. It is worth mentioning that while these are the most common approaches to the order batching problem, they are not the only ones. Several new approaches have been developed in the last few decades, or so that also try to utilise genetic algorithms and other similar metaheuristic approaches, which will not be covered due to the scope of this thesis.

2.2.1 Linear Programming Approach

The first approach is the LP approach, and two reasons make this approach attractive. The first reason is that this approach forces us to model our problem as a mathematical model, which helps us clearly define the problem. However, the second reason, which is even more important, is that an LP solution is global optimum, and this is often desired for industrial engineering. Another way one could perceive the batching problem is like a knapsack problem[3], where some of the LP models have gotten their inspiration, but there were still a few key differences between them. When it comes to modelling order batching as an LP, there have been several attempts throughout the years; one of the first-ever LP models for order batching was created by Vinod (1969), and some of the others that were also popular throughout the years include Kusiak, Vannelli & Kumar (1986), Bozer & Kile (2008) and Kulak, Sahin & Taner (2011).

However, while an LP approach does provide optimal solutions, the order batching problem is known to be an NP-hard problem and is only solvable in polynomial time if the batches contain at most two orders (Briant, et al.,2020). This poses a problem for any case when the batch size is larger than two orders. Another reason is that most LP models do not consider different order priorities but treat them equally. Due to these problems, developing solutions for the order batching problem using LP models stopped being the preferred approach. That is why, in the last decade, the focus has also been turning towards efficient metaheuristic approaches, which are essential for further improvement (Cergibozan & Tasan, 2016).

[3] Knapsack problem – "...a set of entities, each having an associated value, from which one or more subsets has to be selected in such a way that the sum of the values of the selected entities is maximized, and some predefined conditions are respected." (Martello & Toth, 1987) (p.213)

2.2.2 Heuristics Approach

Since the LP algorithms are not practical in real life, the next natural step is to look for a heuristic that will provide us with near-optimum solutions, which are, in practice, already more than satisfactory. Within the heuristics algorithms, two main types can be found: seed and time window algorithms (Il-Choe & Sharp, 1991). Seed algorithms usually take one order (the seed) and try to find another order that could be combined with this one. Several methods of selecting the seed order and the algorithms check if another order can be added to the seed order; Koster, Poort & Wolters (1999) provide a very nice overview of these algorithms and policies.

One of the significant benefits of seed algorithms is that they are fast and can handle large numbers of orders with relative ease. However, their batching is relatively primitive and not very efficient. That is where the time-saving algorithms become relevant. While more computationally intensive, they perform better overall than the seed algorithms (Koster, Poort, & Wolters, 1999). The time-saving algorithms work by calculating the time saving between two orders, which is defined as the time it would take to collect both orders individually minus the time it would take if they were collected. This is then done for every possible combination of the two orders, and the one with the largest time saving is selected. The remaining logic of clustering these orders together later varies per algorithm; however, everyone utilizes this time-saving mechanic in some shape or form. In their paper, Koster, Poort & Wolters (1999) have developed a methodology to decide what kind of heuristic algorithm is appropriate and alternatives in case it is too computationally intensive.

2.3 Routing

The final way to approach the warehousing operations is by developing shorter routes when collecting items. This is the most direct and practical approach out of the three. While the first two approaches help when it comes to warehouse operations by reducing time, if the routing algorithm that is used needs to be fully optimised, all these other optimisations are wasted. The routing itself brings the warehouse design, batching, and zoning together and combines the best of both. One can take three approaches in route construction: LP, DP, and heuristics. Routing aims to make the paths workers must traverse as efficiently as possible since the design and layout are most utilised.

2.3.1 Dynamic and Linear Programming Approaches

Within the warehousing literature, this problem is often referred to as the picker routing problem, and the preferred approach to solving this problem is using DP and LP methods. These methods are preferred over the heuristics because they provide a globally optimal solution. One of the first ever DP algorithms developed for a warehouse was by Ratliff and Rosenthal (1982), whose algorithm has been created for only single-block warehouses[4]. After Ratliff and Rosenthal, several extensions to their algorithms have been created, with some of the most popular being by Cornuejols, Fonlupt & Naddef (1985) and Roodbergen & De Koster (2001). There have also been several LP solutions, which are not as popular as the DP solutions due to their high computational intensity; in their paper, Valle, Beasley & Cunha (2017) provide a different insight into how we can model the picker routing problem which can be particularly useful when mathematically modelling the problem.

[4] Single-block warehouses – are warehouses that consist of only one row of aisles, where you could only enter and exit an aisle at the begining or the end.

However, there is one central assumption that most of these models assume, which is only the case in some warehouses: that they assume a rectilinear layout[5]. When Ratliff and Rosenthal developed a DP algorithm to solve the picker routing problem, they formulated their problem as a variation of a TSP problem. The travelling salesman problem is famously recognisable as an NP-hard problem. However, if formulated as a rectilinear variation of the TSP (there are some differences between the traditional TSP and the picker routing problem that are discussed in Chapter Three), it is possible to solve it in polynomial time, but if this is not the case the problem again becomes impossible to solve in practice. This approach cannot be used for the warehouses that deviate from this layout. Another problem that these algorithms face is that they do not consider something called “aisle congestion” (Koster, Le-Duc, & Roodbergen, 2006), which occurs when many pickers go to the same aisle, thus creating a traffic jam in the warehouse, which reduces the speed at which the pickers move. Since each route is constructed individually, the interaction between the routes is not considered.

2.3.2 Heuristic Approaches

In cases where a DP or LP is not possible to implement or takes too much time, a routing heuristic stands out. Unlike the DP and LP, the heuristics will not provide the optimal solution; however, their very cheap calculation requirements allow them to consider other problems, such as aisle congestion (Koster, Le-Duc, & Roodbergen, 2006). Another benefit of the routing heuristics is that they are a lot more intuitive to warehouse pickers; the reason this is important is that the warehouse pickers sometimes deviate from the optimal route that is provided because it is not intuitive to them, which leads to longer picking time and sub-optimal routes. This problem is solved with heuristics because the output is much more intuitive and thus results in less route deviation (Gademann & Velde, 2005). An overview and a comparative analysis of the most popular heuristic can be found in the paper by Koster, Le-Duc & Roodbergen (2006).

2.4 Conclusion

This chapter has given us better insight into the solutions and approaches to warehousing operations and the benefits and downsides of each. Regarding warehouse layout, the company is currently divided into four main zones: beverages, FMCG, frozen goods, and other household essentials. For the thesis, we will consider only two, beverages and the reset, since these two are only physically divided. However, changing the design of the warehouse or implementing any changes to its layout would require it to pause its operations for a couple of days, which would incur too high costs for this project. That is why our search for the solution was narrowed to two approaches, batching and routing since they will impact the warehouse's performance the most. When implementing the solutions, batching and routing will have to be implemented separately for beverages and the rest.

[5] Rectilinear layout – is defined as a graph where there are nodes set up in a rectangular shape representing the beginning and the end of each aisle and in between two vertical nodes there are items that must be collected.

3 Mathematical Model

When it comes to modelling a warehouse picking problem, several decisions and assumptions need to be made. The assumptions include the layout of the warehouse and others that will be used to relax the constraints of the model to make it solvable. These assumptions will help us in the solution generation phase and algorithm selection.

3.1 Warehouse Layout

The company's warehouses are usually split into two zones: the FMCGs and the bottling section. We will have a separate algorithm running for batching and picker routing. The warehouse layout that we will consider for both types of zones and the problem will be based on the works of Vallea, Beasley, & Cunha (2017) since the real layout of the warehouse can be perceived as a rectilinear layout. A model representation of the warehouse layout can be seen in *Figure 4*.

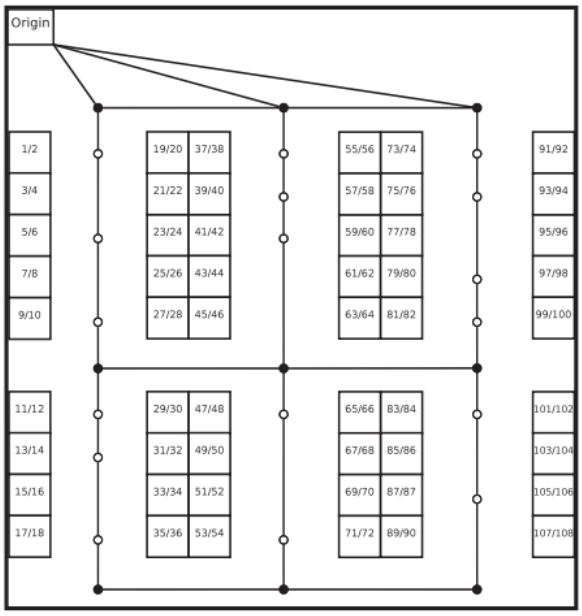

Figure 4: Warehouse layout (Vallea, Beasley, Cunha, 2017)

The following graph shows a section of the warehouse that consists of four blocks, which make a total of three aisles (vertical lines) and three cross-aisles (horizontal lines). The graph consists of two sets of vertices: V_A which represents the ends and beginnings of sub aisles (black dots), V_o which represents the locations of where the items must be collected (white dots).

3.2 Problem Formulation

The warehousing problem can be formulated as follows. There is a set of orders O that we need to collect in a warehouse. Each order $o \in O$ consists of a set of products $p_o \subseteq P$. However, not every product has its own dedicated location. We assume here that every product that is within arm's reach of the warehouse picker, so both sides of the aisle and on any of the shelves, is in the same location. This reduces the number of nodes and the complexity of the computations. From this a new subset of locations $L_o \subseteq L$ for each order $o \in O$, note that the size of the set L_o and p_o does not need to be the exact same size, since there could be multiple products at the same location. Given these sets, our goal is to collect these orders in the warehouse by traversing the shortest distance possible. This

problem will now be split further into two sub-problems, the order batching problem and a picker routing problem. The warehouse picking problem can be solved without batching. However, literature shows that order batching significantly improves the overall performance of the warehouse, which is why we have decided to also include batching in our mathematical formulation.

3.2.1 Order Batching Problem Formulation

Order batching problem can be formulated as follows, given a set of orders $o \in O$ and a set of locations $L_o \subseteq L$ the goal is to batch these orders such that the overall travel distance that the picker must traverse is reduced. For this, we need to make several assumptions:

1. We assume that every order fits in the cart independently.
2. We assume that every picker has a cart with the same capacity C.
3. We assume that the orders will be split upon arrival into two parts the bottles and the FMCGs, each for their respective zones.
4. For the FMCG zone in the warehouse we assume that the orders cannot be split amongst pickers, this means that the picker will have to utilise the sort-while-pick[6] policy when collecting items.
5. For the bottled zone, we assume that the order can be split amongst the pickers and will be sorted once they arrive to the depo; this means that the workers will utilise the pick-and-sort[7] policy while collecting items.
6. Since orders do not have the same priority, we will have to sort them into priority sets based on their deadline.

We will split each order $o \in O$ into two parts (assumption 3) o_f for FMCG and o_b for bottles, thus one order $o = o_f \cup o_b$. Since batching will be done separately for each zone, we can take one of the two as the order thus making two new order subsets O_f, $O_b \subseteq O$ however, the batching model will be the same for both zones. Only difference between the two is that for bottles instead of orders being batched we will be batching a set products $p \subseteq P$ (assumption 5) such that $p = \bigcup_{o \in O} p_o$. It is important to note that for batching in both zones we will not be looking at the batches as a set of products but rather as a set of locations l_o. To be able to formulate this problem mathematically we will need to create set of possible batches S that consists of orders o such that $s \in S$ contains a set of orders $o_s \in s$. From this set, we can then calculate the length of each route of each batch $s \in S$.

However, since not every order has the same priority (assumption 6) we have decided to define three different classes of priority for orders. The set of orders O will be split into three different subsets $o_j \subseteq O$ where j = 1,2,3, with 1 being the highest priority and 3 the lowest. To find an optimal solution we would have to treat each order with an equal level of priority because only then is it possible to find the orders that match best together, but that would collide with our assumption 6. This means that the solution that we will find will not be one that could be considered a global optimum but rather a local optimum. We prioritize optimally batching as many orders from the highest priority class together and filling up the remaining space with orders from priority classes 2 and 3. To implement this approach we will split the order set O into priority classes and into zones (o_{jf}, $o_{jb} \subseteq O$), then we will take the set o_{1f} and o_{1b}, separately, and treat this subproblem as a standard order batching problem where every

[6] Sort-while-pick – is a picking policy where a picker that collects multiple orders, while collecting items sorts them as they collect them into their respective orders.

[7] Pick and sort policy refers to the – is a picking policy where the picker first collects all the items, delivers them to the depo and sorts the items into their respective orders there.

order in that set will have equal priority since the orders are in the same class. Once all the orders have been consolidated, the batches that have remaining space available will be added to the set of orders with the priority class two and treated as orders themselves. The process will be repeated and the order set from class two will be consolidated using the standard order batching problem formulation. This will then finally be repeated for the third class of priority. From the results of this approach, we will be able to define six different types of batches:

- batch with orders only from class one
- batch with orders mixed from classes one and two
- batch with orders only from class two
- batch with orders mixed from classes one and three
- batch with orders mixed from classes two and three
- batch with orders only from class three

To model the consolidation of the set of orders for each class j individually we will use the linear programming formulation of (Muter & Öncan, 2015).

$$min \sum_{s \in S} d_s * y_s$$

s.t.

$$\sum_{s \in S} a_{os} * y_s = 1 \, , o \in O_{b/f}$$

$$\sum_{o \in O_{f/b}} c_o * a_{os} < C \, , s \in S$$

$$y_s \in \mathbb{B}, s \in S$$

Where d_s are length of route of batch s, a_{os} is 1 if order o is included into batch j and 0 otherwise, c_o is the capacity of the order o, C is the total capacity of the cart, y_s is a binary decision variable if batch s is used or not. The set S of possible batches can be generated if all possible combinations of batches satisfy the capacity of the cart constraint, where each batch could be represented as a vector of a_{os}.

3.2.2 Picker Routing Problem Formulation

Once we have found our set of batches that we have to collect and have ensured that it contains all the orders, the next step is to formulate the routes for these batches. To make the problem formulation more understandable we will look at only one batch from the set since the same process will be applied to each batch individually. Before we model the problem, we list several assumptions that have been made:

1. We assume that the warehouse has a rectilinear layout as shown in *Figure 4*
2. Based on assumption 1 we are assuming that a directed weighted graph can be constructed from this rectilinear layout as shown by (Roodbergen & Koster, 2001)

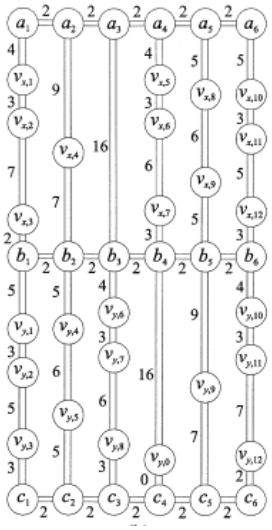

Figure 5: Weighted rectilinear graph (Roodbergen & Koster, 2001)

3. Unlike a standard travelling salesman problem where each vertex can be visited only once here the vertices that belong to the set V_a can be visited multiple times however vertices that belong to the set V_o can be visited only once.
4. Each arc in the graph can be traversed only once
5. We do not consider aisle congestion; that is, every route construction is done irrespective of the previous or future routes to be formed.
6. We assume that there is no route deviation by the pickers

Once we receive one of the batches that need to be considered we split this batch into a set of locations $L_s \subseteq L$ which represents the set of products that must be collected. For the FMCGs we assume that the picker will sort each product into a box with an appropriate order and that they will know which product belongs to which order. Since priority of orders has already been addressed through batching, when it comes to routing every item will have the equal priority when it needs to be collected since only once the entire batch is collected will we deliver it to the depo. That also means that the order will not be picked sequentially but rather every order will be collected simultaneously. To model this problem, we will base our objective function and the constraints on the works of (Vallea, Beasley, & Cunha, 2017). We are given a graph D=(V, A) which has a set of vertices $V = V_a \cup V_o \cup v_s$(starting vertex which can be outside of the rectilinear graph) and a set of arc A which connect these vertices with their respective neighbours. From this formulation the following objective function can be given.

$$\min \sum_{(i,j)\in A} d_{ij} * x_{ij}$$

Where d_{ij} represents the distance between the vertex i and j, which is also the weight of the arc $a \in A$, and x_{ij} which is a binary variable that is 1 if the arc is used in a tour to collect batch $s \in S$ and 0 otherwise. The goal of this objective function is to reduce the distance that the picker has to traverse through the graph to collect all of the items.

Constraints

The constraints used in this model will be similar to the constraints proposed by (Vallea, Beasley, & Cunha, 2017) with one small variation, which is that we do not consider that the model needs to know to which cart which order belongs. In their paper, Vallea, Beasley & Cunha (2017) model the problem with the approach of using a separate trolley for each order and that way, they represent batches in their model. Since they return all the routes at once

for all of the batches, however, we are simplifying our model and providing it only one batch at a time, which is why we do not consider the carts in our constraints. Another thing that we do not consider is the capacity constraints for the carts and the pickers since those are already considered in the batching, so we assume that the batches received already fit into the picker's cart.

Graph constraints

When forming a route there are several graph constraints that need to be considered. (1) the number of arcs going into vertex i ∈ V (δ^-(i)) needs to equal the number of arcs going out of vertex i (δ^+(i)). (2) the total number of arcs that are connected to vertex j and could be used needs to be less than or equal to the total number of arcs that are used for vertex j for every j that is part of the route R. These can be modelled as follows.

$$\sum_{(i,j)\in\,\delta^+(i)} x_{ij} = \sum_{(j,i)\in\,\delta^-(i)} x_{ji}$$

$$\sum_{j\,\in R} g_j \geq\ y_{ij} + \sum_{j,k\,\in\,A(R)} x_{jk}\ \ \forall i \in R, R \subseteq V\ \{v_s\}, |R| > 1$$

$$x_{ij} \in\ \mathbb{B}\ \forall\ (i,j) \in A$$

3.3 Conclusion

In this chapter, the warehouse picking problem was mathematically formulated and divided into two smaller problems. For both problems, a set of reasonable assumptions was provided that made the problem constrained and easier to model. The assumptions and constraints formulated in this chapter will be used in the following chapter to formulate an optimal solution that we plan to implement and use in this thesis. This also ensures that the generated solution addresses the precise problem we are trying to solve for the company.

4 Solution and Implementation

In this chapter, we discuss the solution created for this thesis. This solution consists of two parts corresponding to Chapter Three's problem formulations. In section 4.1, we discuss the batching heuristic that was selected and the reasoning behind this solution choice, after which we briefly explain how the solution was implemented and the structure of the code. Section 4.2 explains how the DP solution has been developed for routing. Here is why this solution has been chosen and what optimisations to the initial model we made are given. In section 4.1, we also briefly explain the implementation and structure of the code.Both solutions have been implemented in a programming language called Julia. While this programming language has not been used throughout our studies, it is a language that is rapidly gaining popularity among industrial engineers within the industry. It is also the company's preferred language.

4.1 Batching Heuristic

For our batching problem, we formulated our mathematical model like an LP, initially driving us towards formulating an LP model as a solution. However, the literature review in Chapter Two shows several downsides to using this approach, with the main one being that the calculation time it would take would not be realistically feasible in real life. That is why a heuristic approach has been chosen to solve the order batching problem.

In their paper, Koster, Poort, & Wolters (1999) provide an excellent overview of seed and time-saving algorithms and a methodology for selecting a heuristic based on different warehousing parameters. When deciding which batching heuristic to select, an important question we need to ask is: What size are the batches? According to Koster, Poort & Wolters (1999), if a batch contains no more than six or seven orders, then a time-saving algorithm performs better; however, if the batch contains more, then a seed algorithm is preferred. In our case, one picker continuously collects one batch at a time, and the picker's capacity is limited by the cart size, which can handle up to 6 boxes (the difference between the boxes and orders will be addressed in the implementation subsection). This makes our cart size an edge case small enough to use a time-saving algorithm.

One downside to the batching heuristics provided in the paper by Koster, Poort & Wolters (1999) is that all the batching heuristics treat all orders as being equal. However, such an approach deviates from the reality of our problem, where the orders are sorted based on what time they have arrived. To relax this constraint in Chapter Three, we have decided to model our batching problem by forming three priority classes based on how urgently these orders need to be delivered. In such a model, we proposed that if the difference between the order arrival times is small enough, the difference in the priority can be ignored. This means that the orders in the same priority class could be treated equally.

If we treat the orders within the same class equally, then the rest of the mathematical model explained in Chapter Three can be easily implemented. The approach used in this solution is almost identical to how batching was explained in Chapter Three. We would first batch as many orders as possible, using a time-saving heuristic, in the highest priority. Once all the batches have been created, we take the ones that still have space and place them into the second priority class, where we treat them as orders whose capacity is the sum of all the orders from that batch. We repeat this process in priority classes two and three.

4.1.1 Priority Policy

After batching, as explained in the section above, we noticed that all the batches can be classified into one of the six types listed in Chapter Three. The problem with this approach is that while the orders have been batched together, we have yet to receive an order in which the batches should be collected. That is why a priority policy has been constructed to streamline the picking process and reduce the order delay time. The prioritization policy always takes the batch with the most orders from the priority set one. If both have the same number of orders from priority class one, then we compare the number of orders from priority class two. If both have the same number of orders from priority classes one and two, we will look at which batch has a greater size. If both batches are identical, then a random one of the two is selected.

When modelling an order-batching problem with strict deadlines, finding an optimal solution becomes tricky. The main reason for that is that the order batching LP solutions always assume that there is an infinite workforce because every batch is immediately collected, and no waiting time occurs. However, this assumption is not realistic in real life. Since the workforce in a warehouse is limited, there are occasions when some orders get delayed due to capacity constraints, which we cannot control. That is why this batching heuristic tries to consolidate as many orders that are about to "expire", and the priority policy always puts these orders at the front of the queue. This way, we, in theory, minimize the order delay compared to the company's current order batching algorithm, which does not consider order priority.

4.1.2 Time Savings Algorithm

In the previous section of the batching heuristic, we provided a high-level description of the solution, how the batching interacts with different priority classes, and how these batches are ordered after they have been created. This section explains which time savings algorithm has been selected and how this algorithm works.

When selecting a specific time-saving algorithm, selecting an appropriate routing heuristic is very important. This is because depending on the selected routing heuristic, the time saving will be calculated differently. There are two reasons for selecting a routing heuristic instead of an optimal routing algorithm: DP or LP. First, it is because, most of the time, it is enough to select a heuristic that approximates the time difference for time-saving. The second reason is that the calculation behind a routing DP or LP is usually longer than that of a heuristic. Since time-saving algorithms calculate a substantial number of routes for their time savings, the speed of route construction significantly impacts performance.

In their paper, Koster, Poort & Wolters (1999) explain how the best time-saving algorithm to use if the batches contain less than six to seven orders is a variation of an algorithm created by Clarke & Wright (1964) and by using the S-shape[8] heuristic for time savings calculations. We have chosen to use the S-shape heuristic due to the high item density within the warehouse, which, after several discussions with the company supervisor and referring to the literature, is the optimal heuristic to be used. In their paper, they talk about three variations of the Clarke & Wright algorithm: the basic variant, the recalculation of the time savings matrix, and the limitation of the number of batches. In this thesis and according to Koster, Poort & Wolters (1999), the algorithm we will use is the Clarke & Wright second variant, which we will

[8] S-shape heuristic – this heuristic looks at all items in a route and check in which aisle they are located it then constructs a route such that it goes through the entire aisle where there is at least one item to be collected it does so following an s shaped pattern, which is where the name comes from. Refer to the appendix for a graphical example.

explain in detail. This algorithm, which consists of five steps, is explained in a paper by Koster, Poort, and Wolters (1999).

1. Calculate the time savings for all possible pairs of orders depending on the batch's capacity. Time savings are calculated by estimating the length of the route using the S-shape heuristic for each order individually and for the two orders combined in one route and taking the difference.
2. Sort the time savings in a decreasing order.
3. Find a pair with the highest time savings; if there is a tie, select one at random.
4. Combine these two orders to form a batch if the picker's capacity allows it; if that is not the case, check the next two orders.
5. If there are more orders that could be batched together, take the current batch and go back to step one, except this time, the batch

4.1.3 Code Implementation

Once we know how the matching algorithm works in theory, it is essential to explain how it was implemented in practice so the company could use it. In section 1.6, we explained how every order is split into boxes; this is important because when an order arrives, we receive a list of products. These products are defined as structs[9] which contain specific fields; these fields contain information such as the product's name, id, volume and location within the warehouse. We then take this list of products and divide the order into several identical boxes. Since every product has a different volume, we place the products into a box until the sum of the products placed in a box does not exceed the box's volume.

However, this still posed a problem of what items to place in which box, which presented us with a standard knapsack problem. To solve this problem, a greedy seeding algorithm has been implemented to sort the items based on their proximity and place as many orders as possible into one box. This way, each box will have its items in one proximity cluster, reducing the route length for each box. Within the code, these boxes are treated as structs that contain the box ID, order ID, from which the items are, and a list of products placed in that box.

[9] Struct – is a data type in Julia that behaves in a similar way as an object in an object-oriented programming (OOP) language with the main difference being that struct cannot have methods that are assigned to them, since Juila is not an OOP language.

Once all orders have been sorted into priority classes, we then divide these orders in their priority classes into a list of boxes, each belonging to their respective priority class. The remaining logic of the solution is the same as described in the previous sub-sections of 4.1, where instead of batching orders, we are batching boxes, which are only partial orders. Since every cart in the warehouse can carry up to six boxes, the code will output a list of structs called batches, which will contain a list of boxes that belong to that batch.

To ensure that the code is correctly running, a list of unit tests has been provided to ensure correct behaviour, as well as one final unit test that tests the entire batching with an example and a precalculated outcome to verify the validity of the code.

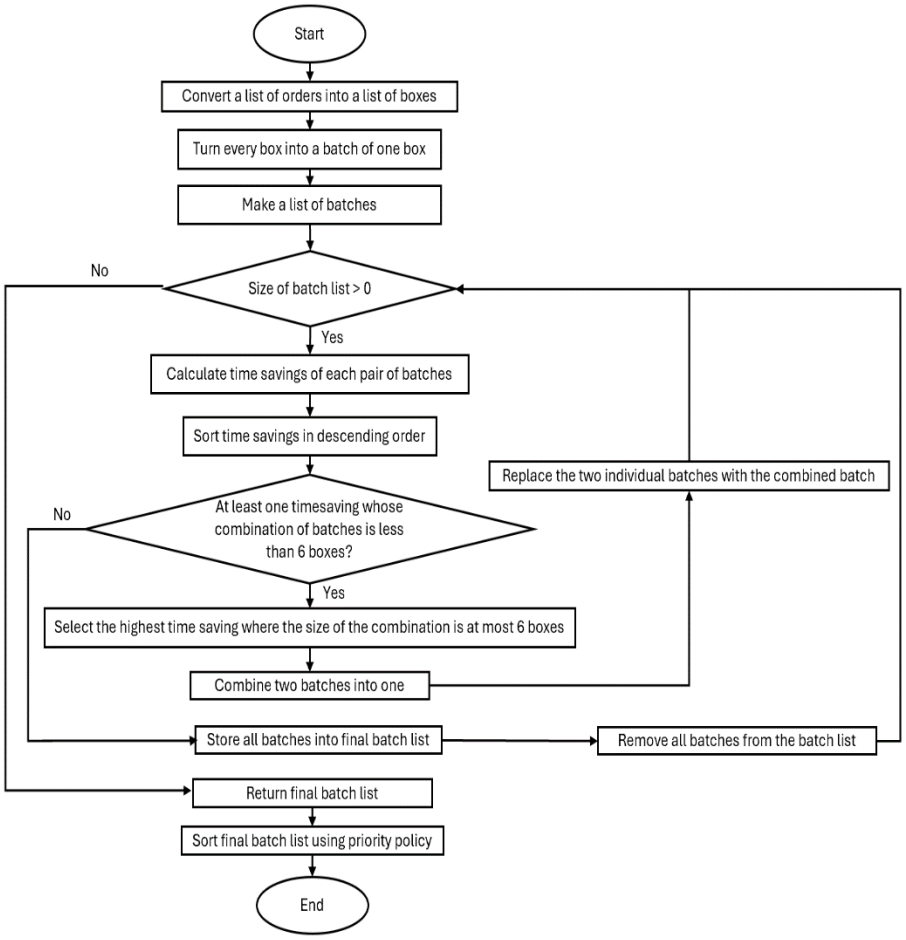

Figure 6: Batching algorithm flowchart

4.2 Routing Dynamic Programme

Once we have consolidated the orders into batches and provided an order for which these batches have to be collected, the next step is to construct routes for each. In Chapter Three, we explained how the layout of the company warehouse is rectangular, and in Chapter Two, we looked at several approaches that we could take to solve the picker routing problem. Route construction is a critical part of warehousing operations. It also has the most direct impact on the order throughput rate, which is why an approach providing an optimal solution has been chosen.

When a picker receives a batch, that batch contains a list of products that must be collected; however, in Chapter Three, we propose that not every product has a unique location. This is why, for the picker routing problem, we must look at the list of locations a batch contains and find the shortest route to visit all these locations. The first ever DP approach that has shown how to solve a picker routing problem in polynomial time was created by Ratliff & Rosenthal (1982); however, in the DP model, they only consider a single-block warehouse, which differs from the warehouse that the company currently has. Almost two decades later, an extension of their algorithm has been made by Roodbergen & Koster (2001); in their extension, they have used the logic of Ratliff's and Rosenthal's algorithm but have extended it to a two-block warehouse. Since the warehouse layout in their paper is like that of the company, the solution generated here has been inspired by their work.

4.2.1 Graph Theory Concepts Relevant to Routing Algorithms

The original DP model created by Ratliff & Rosenthal provided several mathematical proofs as to why their solution is a global optimum. While we will not cover these proofs in this thesis, we must cover some of the terminology related to their work to understand better how their algorithm works. Since their algorithm is the basis of the paper written by Roodbergen & Koster (2001), the definitions and concepts extend to their paper, from which our solution was inspired.

As mentioned in Chapter Three, this problem could be seen as a travelling salesman problem at first glance; however, there is one key difference between the two. In a travelling salesman problem, the goal is to find the shortest Hamiltonian cycle[10]; however, in the picker routing problem, that is not the case; in our graph, there is a set of vertices allowed to be visited multiple times which makes our solution a normal graph cycle. This relaxation of the constraint makes this variation of the travelling salesman problem solvable in polynomial time.

Each time we connect a set of vertices within this DP model, we need a way to record this connection. A method of representing a section of a rectilinear graph is with equivalence classes. These classes consist of two or three types of data; the first type tells us if there is 0, E (even) or U (uneven) number of arcs connected to a single vertex. The second type of data tells us if the graph consists of 0,1, 2 or more components. When we have a subset of vertices that are part of a graph that is not yet connected, the number of sub-graphs in that subset of vertices is defined as several components. The third and final type of data exists only if the number of blocks in a warehouse is greater than one tells us in which block the components are located.

Figure 7 shows a simplified example of a rectilinear sub-graph that connects only two aisles together in a single block warehouse. When constructing equivalence classes, we only focus on the number of arcs connected to the vertices representing the end of an aisle (big circles) and the total number of existing components. In our example, since we have only one block, the third data type telling us where the two components are located is unnecessary. The equivalence class in this example would consist of three parts: equivalency of arcs in the upper end of a single aisle, equivalency of arcs in the lower end of a single aisle and the number of components a graph has. When forming an equivalence class, we always only focus on one specific aisle at a time.

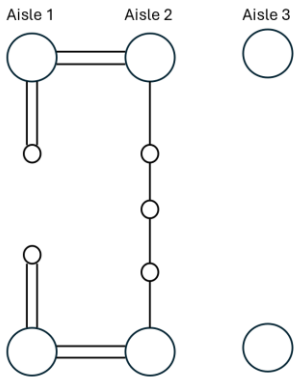

Figure 7: Simple rectilinear graph representation.

[10] Hamiltonian cycle – "Given a graph, find a cycle that passes through every single vertex exactly once…" (Borkar, Ejov, Filar, & Nguyen, 2012)(p. vii)

An equivalence class can differ depending on which section of a graph we are currently looking at. For example, in *Figure 7*, if we focus only on aisle one and ignore the rest of the sub-graph, we will define this equivalence as (E, E, 2C). This means that an even number of arcs are connected to the top and bottom nodes of the aisle and that it consists of 2 components. If we now look at aisle 2, the equivalence class would then be (U, U,1C); since the number of arcs for both the top and the bottom end of aisle two has three arcs, they are therefore uneven, and the total number of components is now one.

4.2.2 Stages, States and State Transitions

Stages

Now that we have a better understanding of certain graph theory concepts, we can use these concepts to understand better the stages of our dynamic programme, what possible states exist, and how we can transition from these states. As a reference, we will use an example of a small warehouse, as shown in *Figure 8*, that consists of two blocks and three cross aisles.

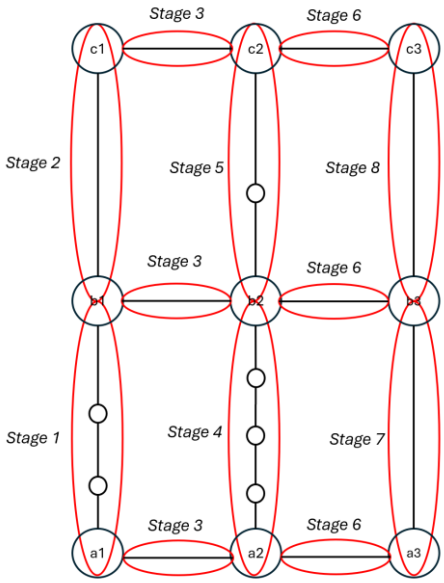

Figure 8: Simple undirected warehouse graph

In every dynamic programme,, there is a set of stages at which we have to make certain decisions. These decisions then define what states are possible in future stages. In *Figure 8*, we can see that red ellipses define the stages of our dynamic program. The order of the stages also shows the order in which the route is constructed. In their paper, Roodbergen & Koster (2001) provide a name for and notation for the different types of stages and how they are then connected. A *final route* in a graph is defined as a tour subgraph T. When constructing a route; we add each subgraph created in a stage j to the tour subgraph T. Roodbergen & Koster (2001) provide three different notations for the stages based on which part of the graph the algorithm is currently at. L_j^- is defined as a subgraph consisting of all the edges left of aisle j and a_j, b_j, c_j. Let L_j^{+y} be defined as the $L_j^- \cup Y_j$ where Y_j is defined as a graph containing all the edges and vertices between a_j and b_j. Lastly let L_j^{+x} be defined as $L_j^{+y} \cup X_j$, where X_j is defined as a graph containing all the edges and vertices between b_j and c_j. To connect this to our example, if we wanted to define stage one using the mathematical notation of Roodbergen & Koster it would be L_1^{+y}, stage two would be L_1^{+x}, stage three L_2^- and so on.

States

To keep track of the current state of the graph at a given stages, we need to use equivalence classes. These equivalence classes will allow us to know the current state of a given aisle at that stage. Unlike the example of the equivalence class given in section 4.2.1, this equivalence class needs to contain more information. Firstly, there we need to store equivalency information of three vertices, a_j, b_j and c_j. Secondly, we need to store the number of components that the graph currently contains. Third, since this is a two-block warehouse, when the number of the given components is two, we need to know which two of the vertices are in one component and which one is on its own. There are three possible combinations for when the number of components is two: a-bc, c-ab, and b-ac. In their paper, Roodbergen & Koster (2001) defined 25 unique equivalence classes, representing all the possible states a graph can have at a given stage. However, not every equivalence class that contains two components tells us how the vertices are distributed between the two. That is because, for every combination of equivalency except E, E, E, it is possible to deduce how the vertices are distributed. Therefore, the third information will be provided only for the equivalence classes that contain E, E, and and E and have two components. For the proofs, we refer to the paper.

State transitions

Two types of state transitions depend on the stage that we are transitioning to, and for every transition, there is a cost that is equivalent to the additional length that is added to the route. If we are transitioning to a vertical stage (L_j^{+y} or L_j^{+x}) then we have six possible choices. Ratliff and Rosenthal (1982) originally defined these six choices in the paper, which contains extensive proof of how those six options were defined. If the next stage is a horizontal stage (L_j^{-}) we have fourteen different choices. Since the number of possible horizontal choices depends on the number of cross-aisles in a warehouse, the possible number of choices was initially less than fourteen in the paper from Ratliff and Rosenthal (1982) but had to be extended to fourteen by Roodbergen & Koster (2001).

Vertical configurations

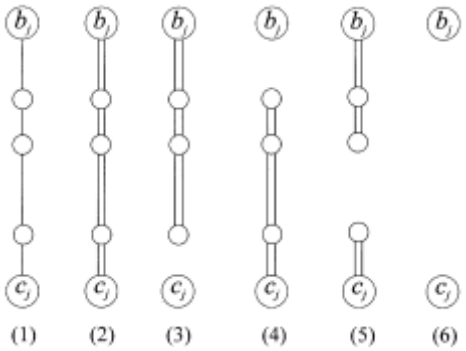

Figure 9: 6 possible vertical configurations (Roodbergen & Koster, 2001)

If we are transitioning from stage L_j^{+y} to L_j^{+x} or from L_j^{-} to L_j^{+y} we must select one of these six options. However, four constraints need to be considered when considering the possible states. Firstly, configuration 6 is only possible if no items are to be collected in that specific sub-aisle. Secondly, if the number of items is less than two, configuration five cannot be selected since configuration five looks for the largest gap between the two item vertices. Since this is a discrete Markov Decision Process, the number of possible states and configurations that can be selected depending on the graph's current state brings us to our third and fourth assumptions. Depending on what stage we are in, there are two different tables, and depending on the current state, a list of possible configurations and the following states are

given. We refer to the appendix for these tables and to the paper of Roodbergen & Koster (2001) for further proof of these constraints.

Horizontal configurations

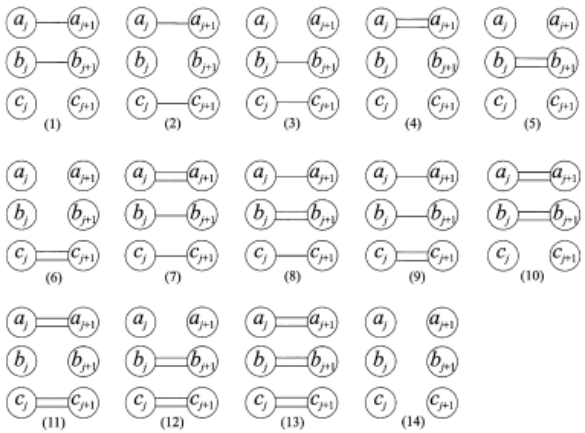

Figure 10: possible horizontal configurations (Roodbergen & Koster, 2001)

In the case that we are transitioning from stage L_j^{+x} to L^-, there are fourteen possible configurations, as shown in Figure *10*. It is important to note that configuration 14 is possible if no items are to be collected right of aisle j. For the other possible configurations, like the vertical stages, there is a table that, depending on the previous state, provides a list of all possible configurations and future states. We refer to the appendix for the table of all possible state transitions.

4.2.3 Further Optimization of the Dynamic Programme

In this thesis, an improvement to the algorithm proposed by Roodbergen & Koster (2001) has been made to reduce the calculation time and complexity. Within this DP algorithm, the goal is at every stage to reach ideally all 25 states from the list of current possible states while removing the duplicates by selecting the cheapest of the two. However, it is not always possible to reach all 25 states. For example, in stage one, we start from the state (0,0,0,0). Since there is only one current state at stage one, we can have at most five options from the five configurations, assuming the sub-aisle is full. From this approach, we are trying to generate as many unique states as possible at every stage since they cover more possibilities. In this thesis, we propose that there is a specific case where exploring certain configurations will never lead to an optimal solution and could, therefore, be excluded.

Picture a simple three aisle single block section of the warehouse as shown in *Figure 11*. In this warehouse section, there are three aisles with items located in aisle one and three. If we followed the algorithm from Ratliff & Rosenthal (1982) in stage one, we would find that there is a total of five possible configurations to be chosen, which leads us to five possible states; in stage two, we would have four possible options according to Ratliff & Rosenthal (1982) paper where they present five horizontal configurations to select from.

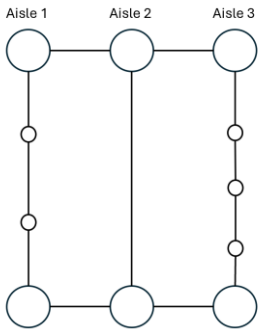

Figure 11: Example of a single block warehouse

This leads us to stage three, where, since no items are to be collected from that aisle, we have three possible configurations to choose from. Configurations 1, 2, and 6 are the same as configuration 2, and configuration 5 is not possible since the number of items is less than two. However, in such a scenario, selecting configurations one or two will never lead to an optimal solution.

Proof

In stage 1, we could select five configurations which lead to 5 possible states, which are (U, U, 1C), (E, E, 1C), (E, 0, 1C), (0, E, 1C), (E, E, 2C). In stage two, when selecting horizontal configurations, we must consider all the previous five as possible current states. After trying all the possible configurations on all possible states and removing the duplicates, we get the following possible states at stage two: (U, U, 1C), (E, 0, 1C), (E, E, 2C), (0, E, 1C), (E, E, 1C). Once we reach stage three, since there are no items to be collected on each of these five possible states from stage two, configuration six can be used, and since no arcs are created, the total cost of this configuration is zero. This means it is only optimal to use configuration 1 or 2 in stage three if they generate a new state. If we look at Table 2 from Ratlif and Rosenthal (1982), we will see that they will never generate a new state; therefore, configuration 1 or 2 will never be optimal because the same states can be achieved with zero costs.

This will be true for any arbitrary empty aisle in a single-block warehouse. To extend this proof further for the two-block warehouse, it can be shown that for any arbitrary sub-aisle, it is possible to achieve the same number of unique states by using just configuration six if no items are collected in that sub-aisle. The logic of the proof would be the same, except that there would be more possible states for which the proof should cover.

.

4.2.4 Numerical Example

To show how this algorithm works in practice, a numerical example has been provided that shows stage-by-stage results for further understanding. In this numerical example, we will take the graph from Figure 8 and turn it into an undirected weighted graph, where the weights represent the distances between the vertices, as can be seen in Figure *12*.

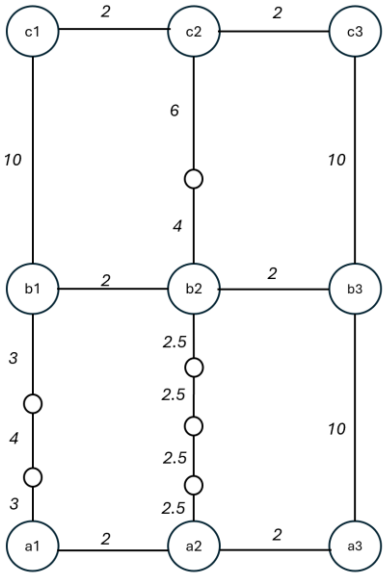

Figure 12: Weighted undirected warehouse graph

Using the information from this graph, we will create three tables that show the algorithm's stage-by-stage results. The tables will contain information about which state we started, what configuration was used, what the next state is, and what the current cost is.

Stage 1				Stage 2			
From state	Config.	Next state	Cost	From state	Config.	Next state	Cost
(0,0,0,0)	**1**	**(0,U,U,1)**	**10**	**(0,U,U,1)**	**6**	**(0,U,U,1)**	**10**
(0,0,0,0)	2	(0,E,E,1)	20	(0,E,E,1)	6	(0,E,E,1)	20
(0,0,0,0)	3	(0,E,0,1)	14	(0,E,0,1)	6	(0,E,0,1)	14
(0,0,0,0)	4	(0,0,E,1)	14	(0,0,E,1)	6	(0,0,E,1)	14
(0,0,0,0)	5	(0,E,E,2)	12	(0,E,E,2)	6	(0,E,E,2)	12

Table 1: Numerical example solutions pt. 1

Stage 3				Stage 4			
From state	Config.	Next state	Cost	From state	Config.	Next state	Cost
(0,U,U,1)	**3**	**(0,U,U,1)**	**14**	(0,U,U,1)	**1**	**(0,E,E,1)**	**24**
(0,E,E,1)	12	(0,E,E,1)	28	(0,E,0,1)	1	(0,U,U,1)	28
(0,E,0,1)	5	(0,E,0,1)	18	(0,E,0,1)	3	(0,E,0,1)	33
(0,0,E,1)	6	(0,0,E,1)	18	(0,E,0,1)	4	(0,E,E,2)	33
(0,E,E,2)	12	(0,E,E,2)	20	(0,0,E,1)	4	(0,0,E,1)	33

Table 2: Numerical example solutions pt. 2

Stage 5				Stage 6,7,8			
From state	Config.	Next state	Cost	From state	Config. 6 | 7,8	Next state	Cost
(0,E,E,1)	1	(U,U,E,1)	34	(U,U,E,1)	14 | 6	(0,0,0,1)	34
(0,E,E,1)	2	(E,E,E,1)	44	(E,E,E,1)	14 | 6	(0,0,0,1)	44
(0,E,E,1)	3	(E,E,E,2,a-bc)	36	(E,E,E,2,a-bc)	14 | 6	(0,0,0,2)	36
(0,E,E,1)	**4**	**(0,E,E,1)**	**32**	(0,E,E,1)	14 | 6	(0,0,0,1)	32
(0,U,U,1)	1	(U,E,U,1)	38	(U,E,U,1)	14 | 6	(0,0,0,1)	38
(0,U,U,1)	2	(E,U,U,1)	48	(E,U,U,1)	14 | 6	(0,0,0,1)	48

(0,U,U,1)	3	(E,U,U,2)	40	(E,U,U,2)	14 | 6	(0,0,0,2)	40
(0,U,U,1)	4	(0,U,U,1)	36	(0,U,U,1)	14 | 6	(0,0,0,1)	36
(0,E,0,1)	1	(U,U,0,1)	43	(U,U,0,1)	14 | 6	(0,0,0,1)	43
(0,E,0,1)	2	(E,E,0,1)	53	(E,E,0,1)	14 | 6	(0,0,0,1)	53
(0,E,0,1)	3	(E,E,0,2)	45	(E,E,0,2)	14 | 6	(0,0,0,2)	45
(0,E,0,1)	4	(0,E,0,1)	41	(0,E,0,1)	14 | 6	(0,0,0,1)	41
(0,E,E,2)	1	(U,U,E,2)	43	(U,U,E,2)	14 | 6	(0,0,0,1)	43
(0,E,E,2)	2	(E,E,E,2,c-ab)	53	(E,E,E,2,c-ab)	14 | 6	(0,0,0,2)	53
(0,E,E,2)	3	(E,E,E,3)	45	(E,E,E,3)	14 | 6	(0,0,0,3)	45
(0,E,E,2)	4	(0,E,E,2)	41	(0,E,E,2)	14 | 6	(0,0,0,2)	41
(0,0,E,1)	3	(E,0,E,2)	45	(E,0,E,2)	14 | 6	(0,0,0,2)	45

Table 3: Numerical example solutions pt. 3

These tables show the results of all the unique classes that could be generated at each stage and the associated cost. Since there are no items to be collected in the last aisle after we have constructed stage five, it is unnecessary to continue constructing the route since there is no need to traverse the remaining aisles. Nevertheless, the results for stages six, seven and eight were provided for mathematical completeness of the solution for stage 6. We would have selected configuration 14, which would make the next state always the same since stages seven and eight would use configuration six, which does not change the previous state nor add additional costs.

In stage five, several solutions are marked red. Since stage 5 was the last stage where additional arcs would be added to a graph, any solution whose sub-graph contained more than one component could not be turned into a cycle, therefore not making it a valid solution. Once all the possible and infeasible solutions have been filtered, the subgraph with the lowest cost is selected and traced back through previous stages. The optimal solution for this example is marked in bold at each stage. Once we know which configuration to select at each stage, an optimal route can be constructed, as seen in *Figure 13*.

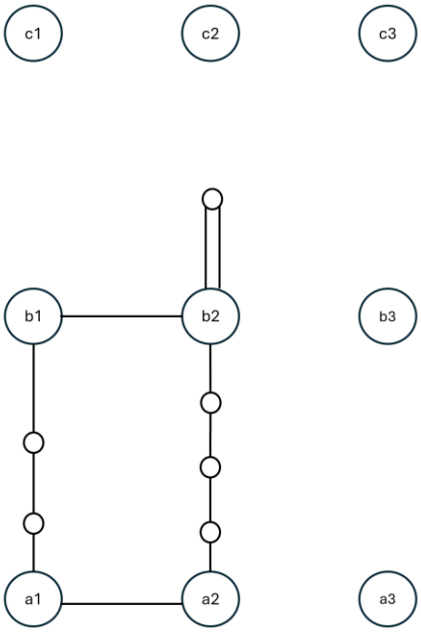

Figure 13: Optimal route solution

4.2.5 Code Implementation

Before explaining how this algorithm has been implemented into code, it is important to understand which data structures have been selected and how the data has been organized. A location matrix has been created to store the graph data, such as each item's distances and locations. The reason behind selecting a matrix to store data rather than a graph is that sorting and accessing the data in a matrix is much faster than in a graph and requires fewer overhead costs. Each item location was stored as a tuple that contained three number aisles, sub aisle (1 for bottom and 2 for top) and the distance from the bottom end of that sub aisle. Once the algorithm receives a batch, it gathers the list of all the locations from a batch and sorts them into the location matrix, where columns represent the aisle, and rows represent sub-aisles. Each element of the matrix would contain a list of distances from the bottom end of that sub-aisle that the picker would need to visit.

Another critical decision was to select how the state transition tables would be stored to have quick access to the possible states and configurations. For these tables, three dictionaries were created with respect to the tables, where the key was the current state, and the value was a list where the index would be the configuration and the element of that index would be the next state. The reason behind selecting the dictionary as a data structure was because, in Julia, depending on the type of data and the amount of data, the dictionary will select an optimal low-level data structure optimized for quick access.

Once the data has been sorted into a location matrix, an algorithm will calculate all possible combinations of states at each stage, remove the duplicates, and store the results of that stage in a matrix. The solution matrix consists of states with four fields: length, which represents the cost; previous class, configuration, and the current class. The solutions of each stage are placed in one column of a matrix, which has a dimension of 25 by the number of stages. Once generated, the algorithm returns the solution matrix from which the route can be constructed.

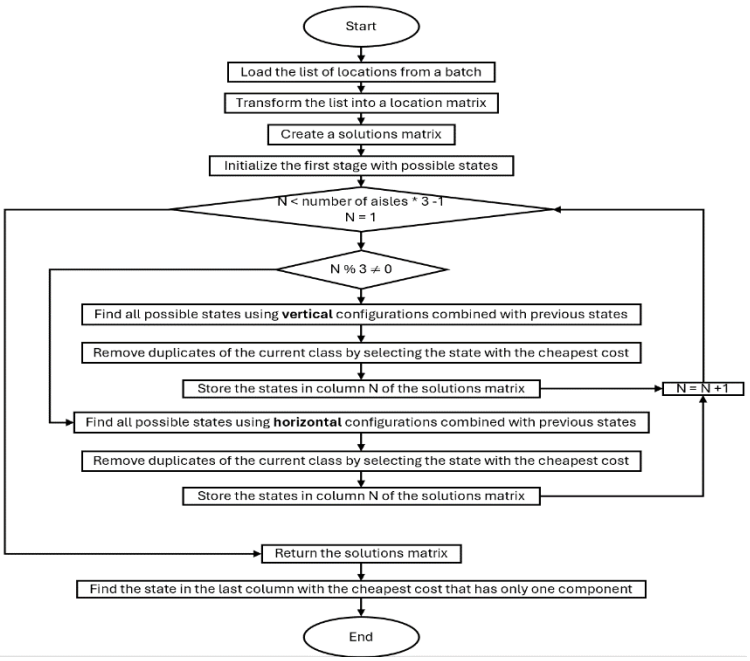

Figure 14: Routing algorithm flowchart

4.3 Conclusion

In this chapter, we have discussed two solutions created for this thesis and the mathematical background necessary for a deeper understanding of both. Aside from the background of how they were created, we also showed how the routing algorithm could be improved to reduce the necessary calculation resources. A high-level explanation of the code and key design decisions has been provided, providing insight into both solutions' implementation. Using the solutions developed in this chapter, we plan to compare the solution created here with the one the company currently uses and do a performance analysis where we discuss the differences between the two in terms of the KPIs selected in Chapter One.

5 Performance analysis

This chapter discusses the performance of the company algorithms and the algorithms developed in this thesis. The algorithm's performance will be based on some KPIs selected in chapter one. The chapter consists of three main sections. 5.1 discusses the setup necessary for comparing the algorithms and explains how the company algorithms have been implemented. Section 5.2 will discuss the results of the analysis, what variables were changed, and how they affect the performance of the algorithms. Section 5.3 discusses the analysis's conclusions and the results' validity.

5.1 Experimental Setup

We have created a simulation environment to conduct the performance analysis between the routing dynamic program and the batching heuristic developed in this thesis against the company routing and batching heuristics. The simulation environment that we have created can be modified based on several input parameters that we change throughout our experiments. The first step in constructing a simulation environment is creating the warehouse. For the construction of the warehouse, the rectilinear layout assumed in Chapter Three is used. In each experiment, the number of aisles can be changed, thus changing the size of the warehouse; the number of cross aisles remains constant at three.

As the number of aisles increases, the number of products increases linearly. This means that for our experiments, every new aisle added increases the number of products by a fixed amount. These products are then evenly distributed amongst the aisles. This behavior was based on observing the company warehouse in Munster, where the number of products remains roughly the same in each aisle.

When it comes to the total number of products that a warehouse has, we conducted several iterations of the simulations where we kept all of the variables constant except the number of products. We noticed that the larger the number of products, the more consistent the simulation outputs become. The main reasoning behind this is that as routes are being constructed, the more locations we have, the larger the dataset of possible scenarios covered; thus, when measuring the distances of the routes, the average distances seem to converge better. This number was around 1000 products per aisle.

After the warehouse has been defined with the number of aisles and the number of products, the next step is to simulate the arrival of orders. This step is crucial since the arrival of orders affects how the batching algorithms will perform and, therefore, will also affect the length of the routes. When it comes to generating the orders to compare the batching and routing algorithms performance, we have decided to look at the warehouse from a static point rather than a dynamic one. What we mean by static is that we generate a set of orders containing a list of products, an ID of an order, and the time to delivery; this time to delivery is between 1 and 120 minutes. The reasoning is that at any point in the warehouse, the newest order can last at most 120 minutes until delivery. This simulation approach is called static because all of the orders are generated at the beginning, and we know from the start which order has to be delivered and where, which could be perceived as if we took one moment in the warehouse in the day, froze it, and tried to solve it using different algorithms. The reason behind choosing this approach over the dynamic simulation is that to simulate the dynamic scenario; we would have to implement time steps where we "freeze" the warehouse situation every n minutes, solve it using batching and routing algorithms, and then, in the next time step we update the order set with the new orders that have arrived and remove the ones that have been fulfilled. This would essentially be running the static simulation many times over.

Lastly, once the simulation environment is set up and the orders are generated, the next step is to select the batching and routing algorithms. In our case, we can choose between batching and routing. For batching, we have the time-saving heuristic created in this thesis and the company batching heuristic; as for the routing, we can choose between the dynamic programming algorithm and the company routing heuristic. The remainder of this section explains both company algorithms in more detail.

5.1.1 Batching

As explained in Chapter One, the batching algorithm utilizes the FIFO policy. This means that when implementing this algorithm, we first had to divide every order into a box; for the division of order into boxes, a seed algorithm explained in section 4.1.3 was reused to place items across different boxes. Since every box belongs to exactly one order, each box will have a deadline that represents the deadline for one order. Once a list of boxes has been created, the oldest box in the list will be selected as a seed. To select the boxes that will be selected for this batch, a centre of gravity heuristic[11](COG) is used. Once a list of batches is made, they are fulfilled in the order they were created.

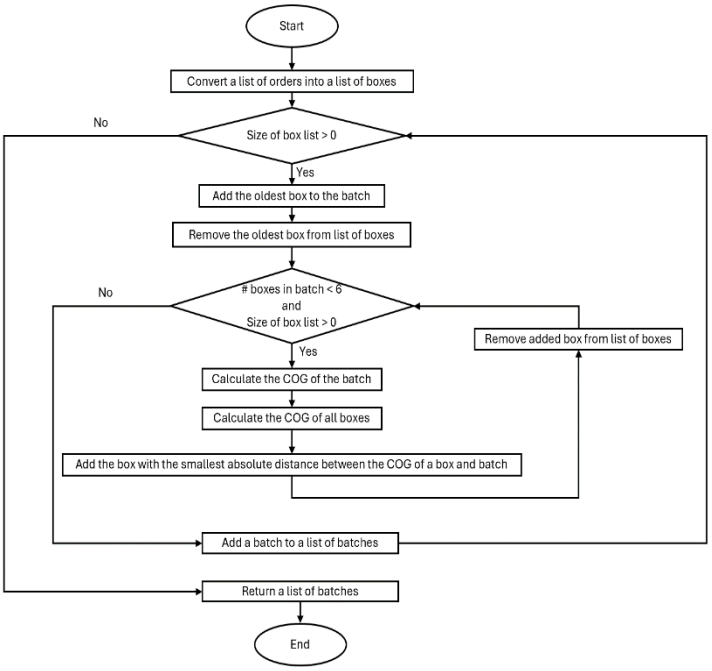

Figure 15: Company batching algorithm flowchart

[11] Centre of gravity heuristic – in their paper Koster, Poort & Wolters (1999) explain that this is a heuristic algorithm used in batching seed algorithms that calculates the gravity of the ever order, which is the average aisle of all the locations in an order, and then looks for the centre of gravity closest to the one of the seed order and repeats this until the capacity of the batch is fulfilled.

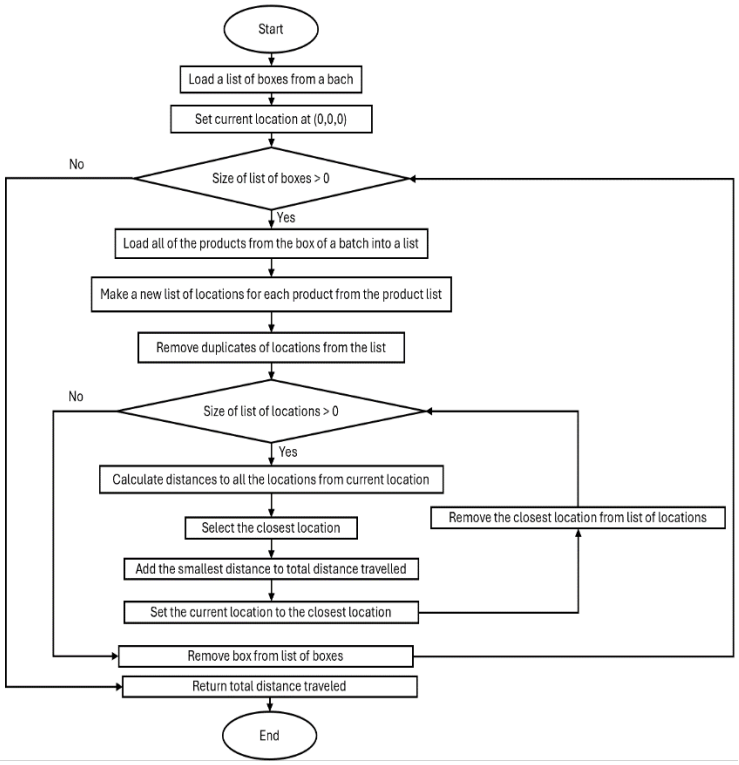

Figure 16: Company routing algorithm flowchart

5.1.2 Routing

Once the batches are created, they are sent to the routing algorithm to construct the routes. As explained in Chapter One, when the routing algorithm receives a batch, the algorithm will extract a list of locations from one of the boxes and create a route using the nearest neighbour heuristic. During the route construction, the nearest neighbour heuristic works by calculating the shortest distance from the current location to the location of every other product. When all the distances have been calculated, the shortest one is selected, and the new current location is the product's location to which it was selected. This product is then removed from the list of locations to be visited, and the process continues until no locations are visited. Once all the items that belong to the first box have been collected, a new route is constructed for the following box from the last location of the picker. This process is repeated until the last box has been collected.

5.2 Comparative Analysis

With the experiment setup described in the previous section, a comparative analysis can be performed where we take the two batching and two routing algorithms and see how they perform in various situations that will be simulated. This analysis can be divided into two main approaches: individual performance and combined performance. In the individual performance, we first batch the orders using the thesis algorithm and then create routes for these batches using the routing algorithm created in this thesis. Then we do the same, except that, in this case, we use the batching and routing algorithm that the company currently uses. This way, we compare the performance of the entire warehouse using two different solutions.

However, because two algorithms are used simultaneously, this could impact the validity of the analysis since it could be possible that only one of the two algorithms could be bad and would, therefore, hinder the performance of the other. That is why a second approach is necessary, where we combine the company's batching algorithm with the thesis's routing algorithm and the thesis's batching algorithm with the company's routing algorithm. By analysing the results of the individual and the combined approach, we can then have a more detailed overview of the algorithms and their performance.

In this comparative analysis, the batching and routing algorithms are only two of many variables that can be changed. That is why we will also look at how these algorithms behave as the warehouse size increases and the number of products in the warehouse increases. This way, we can see how the scale affects the performance of the algorithm and at what rate the route length increases.

5.2.1 Individual Performance

This section compares the performance of doing batching and routing using the company algorithms against the batching and routing using the algorithms developed in this thesis. To compare the two algorithms, we have decided to simulate how the algorithms behave as the number of aisles and products in those aisles increases. We start with ten aisles and 10,000 locations in a warehouse; the reason for such a high number of locations is that it provides us with a larger set of possible scenarios, thus making the results more statistically significant. We then increment the number of aisles by 10 for each run and the number of locations by 10,000 until we reach 100 aisles. The reason for increasing the number of locations linearly with the number of aisles is that it is logical to assume that as the size of the warehouse increases, so does the number of locations. For the number of orders, we have extrapolated from the company data that the number of orders a warehouse receives in a day is approximately around 1000. This would represent the total number of orders in one day in a warehouse, which is constant throughout all the iterations. The main reasoning behind this decision was that if the number of orders were different for every run, the sample size would be different, which could impact the validity of the results.

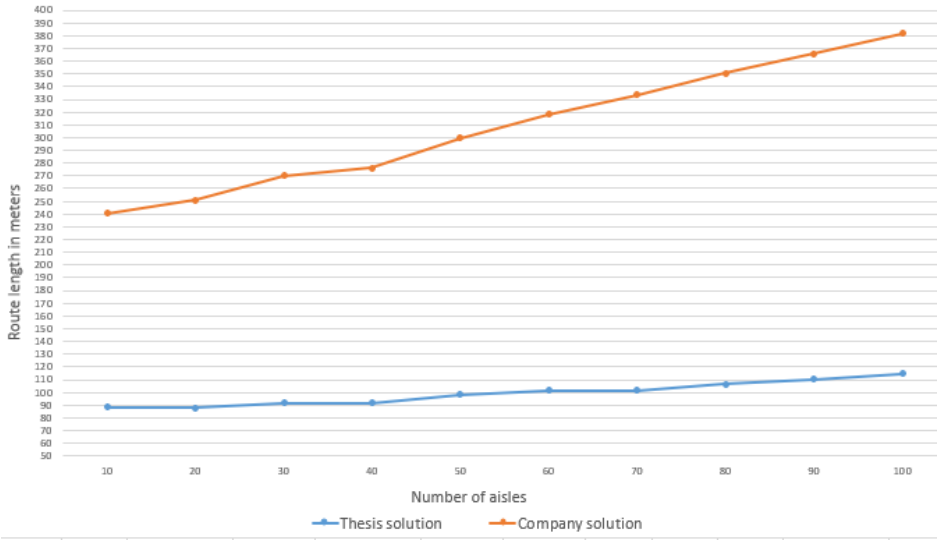

Figure 17: Average length of a route for the individual performance

In *Figure 14*, we can see one of the KPIs from Chapter One, the average length of a single batch, in meters, for the two combinations mentioned above. This graph shows that the overall performance of the batching and routing algorithms developed in this thesis outperforms the current company solution in the length of a route by 67.5%. Aside from reducing individual routes, this graph also shows how much more scalable the thesis solution is by observing the rate at which the route lengths increase as the size of the warehouse increases. While both solutions have linear growth, the company solution has a much higher coefficient than the thesis solution.

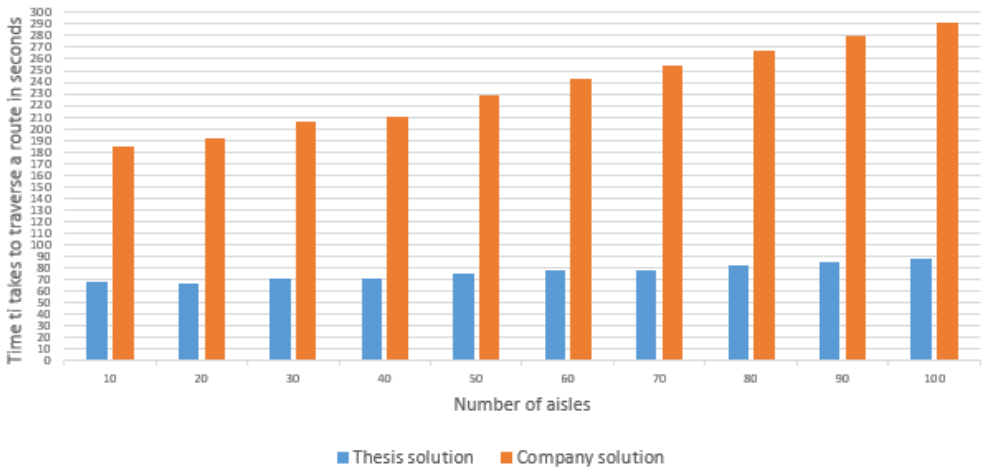

Figure 18: Average time of a route for the individual performance

Aside from just comparing the length of a route, we have also decided to calculate the average time it takes to traverse a route generated by both solutions. Since we do not have the average walking speed of a warehouse worker, we decided to take the average walking speed of an ordinary person, which is 1.31 meters per second (Murtagh, Mair, Aguiar, Tudor-Locke, & Murphy, 2021) and use that as an estimation for the time it would take to traverse the route. It is important to note that this was a rather conservative route estimation since we do not know how long it takes to place the items into a box and unload them at the depot. Nevertheless, a pattern similar to the one observed in the average route length can be seen.

5.2.2 Combined Performance

After doing the individual analysis to improve the validity of our results, simulations using a mix of company and thesis solutions have been conducted where we combine the company batching algorithm with the thesis routing algorithm and the thesis batching algorithm with the company routing algorithm to see the performance. The remaining setup of the experiment variables has remained the same as in the analysis of the performance of the individual solutions. The reason is to keep the simulation environment for the algorithms the same so that nothing except the algorithms themselves affects the performance.

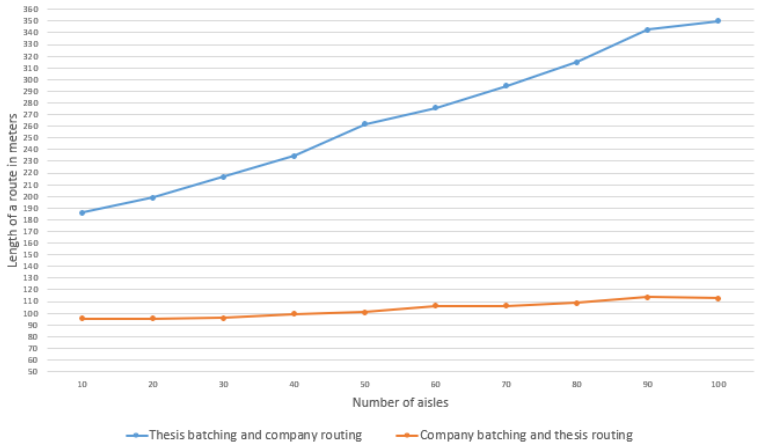

Figure 19: Average length of a route for the combined performance

Figure *16* shows that combining the company batching we conducted algorithm with the thesis outperforms the thesis batching, combining routing algorithm. The overall difference in average length between the two is 60.12%. From this observation, we can conclude that the routing plays a significant role in the performance of the warehouse, and the length of the route's pickers must traverse.

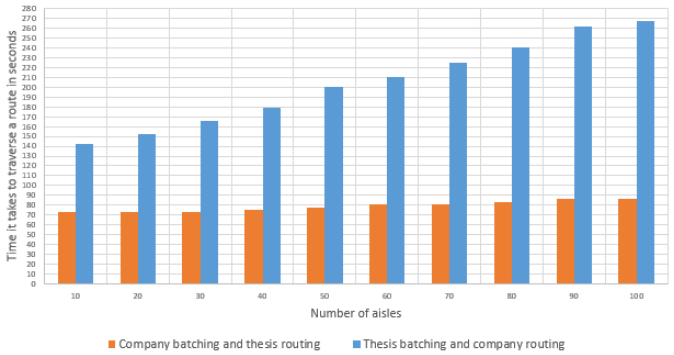

Figure 20: Average time of a route for the combined performance

For the average time it takes to traverse a route, an identical trend can be observed as in the average length of a route. It is important to note again that the estimation here is also conservative due to the same reasons as mentioned in the individual performance.

5.3 Conclusion

After comparing the individual solutions and the combinations of company solutions with the solutions created in this thesis, we can make several conclusions.

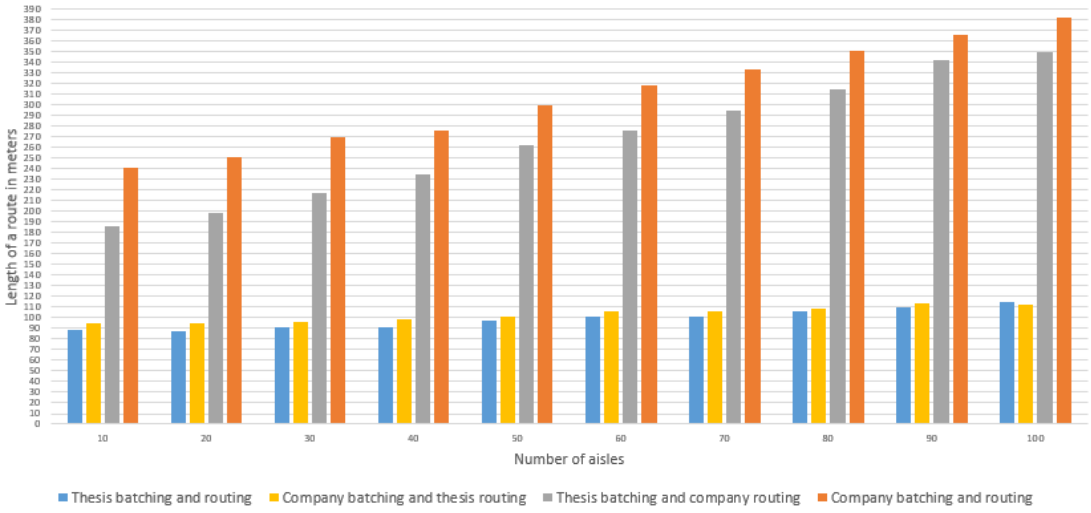

Figure 21: Comparison of the average route length for all four scenarios

From *Figure 18*, the thesis solution had the best-performing algorithms, closely followed by the combination of company batching and thesis routing algorithm. This leads us to conclude that batching is important and can play a significant role, as seen when looking at the performance of the company solution and the combination of thesis batching and company routing. What makes the difference is the routing, which best utilises all the batching and zoning opportunities that those algorithms create. This is also confirmed in *Figure 18*, where the route length drastically drops if a thesis routing algorithm is used.

Through this chapter, we have generated our data, created a simulation environment, and performed multiple experiments with different routing and batching algorithm combinations. Concern about the validity of such an approach could be raised since the generated data and the simulation environment created is not real-world data. However, when comparing the performance of algorithms, real data is optional for a reliable analysis to be concluded. Firstly, all the data, while artificial, has been inspired by the real-world data and information obtained by the company. Secondly, the layout of the warehouse and the way that the locations are stored is based on the real-life warehouses that the company uses. Lastly, when comparing two solutions, we mainly compare how efficient the routes created are. While real-world data would provide more accurate route lengths, the actual performance of the algorithms is scalar, as in, if the routes created by the DP are 67.5% shorter than the one made by the company heuristic, then even if the routes were shorter or longer, the percentage difference should stay roughly the same. This makes the conclusions in this chapter still valid even if the values differ from reality.

6 Conclusion and Future Improvements

In this chapter, we discuss the concluding remarks of this thesis and the future improvements that could be implemented in the company. Section 6.1 answers the main research question and sub-research questions, providing a clear summary of the research and discoveries found in this thesis and an explanation of how the company can benefit from this work. In section 6.2, a list of future improvements to the solutions developed in this thesis and further improvements that could be made to the warehousing operations are also discussed.

6.1 Conclusion

At the beginning of this thesis, we were provided with a problem from the company: the order throughput rate at its warehouses was lower than desired. Through an extensive analysis of the company, we have narrowed down the problem to the warehouse picking process, from which our main research question has revealed itself: *How to reduce average order picking time in the warehouse picking process of Flaschenpost?* However, this question is rather complicated to answer on its own. That is why a list of sub-questions has been created to provide a more structured approach to the answer. The remaining part of this section will answer each research sub-questions, except question six, which will be answered in section 6.2.

Research question 1: What are the current methods and models being used in the company for the picking process?

The warehouse picking process can be looked at from three different aspects: the design of the warehouse, the batching and the zoning within a warehouse and the picker routing problem. For the company, redesigning a new warehouse storage assignment was too expensive to implement, which is why the dedicated storage approach that the company currently uses is considered good enough. This leaves us with batching and routing algorithms. These two aspects are the ones that impact warehousing operations the most and are the parts that we have decided to focus on the most within this thesis. The batching algorithm that the company uses is a greedy seed algorithm, where for the seed selection, they use the FIFO policy, and for finding boxes that will be added to their batch, they use the centre of gravity heuristics to find the orders that are closest to the seed order. Once the orders have been batched, the routing algorithm then proceeds to formulate routes for each box in that batch individually using a nearest neighbour heuristic. Using the algorithms mentioned here creates sub-optimal batches and routes that lead to longer picking times and, therefore, a lower order throughput rate.

Research question 2: What models and heuristics currently exist that deal with the warehouse picking problem?

To answer this research sub-question, we have conducted a literature review from which a theoretical framework, explained in Chapter Two, has been created. The literature review overview written by Koster, De-Luc, & Roodbergen (2006) provides a list of possible solutions, a detailed analysis of each approach, and the papers provide further information about the approaches. From there, we can distinguish three different approaches for batching and routing. For the order batching problem, the three solutions that exist are the LP approach, the machine learning approach, and the heuristic approach. Finally, for the picker routing problems, the three main approaches that exist are the LP, DP, and a heuristic approach. There are benefits and drawbacks to each of these approaches, for which a detailed analysis has been provided in Chapter Two of this thesis.

Research question 3: Which KPIs are relevant for the improvement of Flaschenpost's order-picking efficiency?

To measure the quality of the solutions created in this thesis, we measured the average length of a single batch route and the average time it takes to traverse it. These two were selected as the main KPIs used to measure the performance of the company algorithms and the algorithms created in this thesis. Apart from the two KPIs we have come up with four additional ones that, due to confidentiality or technical reasons, we could not measure but could be helpful to the company should they choose to implement the solution created in this thesis in their warehousing operations. The four additional KPIs are order throughput rate, average backorder rate, total time spent in order fulfilment and order delay time; the reasoning behind the selection of these KPIs and what insights they provide is explained in Chapter One.

Research question 4: How can these algorithms that exist be modified such that they are applicable to our picking process?

Once we identified the problem, answered Research Question 1, found the list of possible solutions, and showed it in Research Question 2, the next step was to define the problem mathematically and select the best approach for this thesis. In Chapter Three, a detailed mathematical problem formulation is provided, which is then used to filter out the solution approaches that were found in Chapter Two. In this thesis, for the order batching problem, a time-saving heuristic was selected for the batching algorithm by Koster, Poort & Wolters (1999), and a prioritisation policy was created to account for the order deadlines. As for the picker routing problem, a DP algorithm by Roodbergen & Koster (2001) was selected and further optimised for faster calculations, as explained in Chapter Four. Once selected, these algorithms were implemented in Julia, which could later be used in the company's analysis.

Research question 5: How to evaluate the mathematical models and heuristics and its impact on the Flaschenpost' s warehouse picking process?

Several more components had to be added to compare the performance of the solution created in this thesis with that of the company. We had to gather data to get some KPIs we found in Research Question 3. Since we could not get the data or the algorithms from the company, for this thesis, the algorithms had to be recreated, and the company data had to be generated. The company algorithms were used as a benchmark algorithm for our performance analysis, and the generated data was used as a simulation environment for the algorithms. To create these experiments, a simulation had to be built to generate orders for products and warehouses, and the routing algorithms would then construct their routes. We refer to Chapter Five for a more detailed explanation of how the experiments were set up and how the data was generated.

6.1.1 Contribution of the Thesis

Throughout this thesis, we have located the exact problem in the company warehouse, identified a list of possible solutions, selected the best one, implemented it, provided a detailed performance analysis of the solution, and answered all the research sub questions. However, in this subsection, we explain what contributions this thesis has generated for the company and how they can benefit from these results.

When comparing the routing and the batching solution generated in this thesis against the heuristic algorithm that the company has used so far, the analysis tells us that the thesis solution creates 67.5% shorter routes than the current company solution. For the company, this means that by implementing this solution, they gain two main things: a higher-order throughput rate, which is what we have defined as the action problem of the company and an

increase in the capacity of their warehouses. Alongside the algorithm, the company can use the information in this thesis and the code documentation to make a list of future improvements to optimize their warehousing operations even further.

6.2 Future Improvements

In this section, we discuss the next steps that the company should take to use the solutions from this thesis, how the solutions can be improved further, and what additional steps the company can take to improve their warehousing operations even further. These improvements are based on the literature used in this thesis and the experience and knowledge obtained by doing this thesis.

If the company wanted to implement this solution into their software, the first step would be to clean and organise the warehouse data. Since the company has over thirty warehouses, it would need to get the layout of each warehouse and the locations of all the products in every warehouse. The location of each product should be stored as a tuple containing the aisle, sub aisle and the distance from the lower end of the sub aisle. Aside from this, each product should be treated as an object with a unique ID and volume. The remaining information, such as the capacity of each box and the workers' speed, should also be estimated. Once we have all this data, we must implement batching and a routing algorithm. Since the orders arrive through the day, this poses a problem of creating optimal batches with the orders currently in the warehouse. However, the solutions generated quickly become obsolete as new orders arrive. To solve this problem, we recommend that the day be split into small time intervals and that new batches be created each time interval; this way, new batches will be recalculated more often, thus keeping the batches up to date. As for the routing algorithm, to reduce the calculation resources, the routes will only be constructed as the pickers take on them; this way, we do not calculate the route of every batch but of only the ones that get collected.

However, as with all optimisation problems, the solutions can constantly be improved. If the company wants to implement the solutions, we provide these further improvements. Since the routing algorithm created in this thesis is a DP solution that finds optimal routes for large warehouses rather quickly, the improvement should focus on the batching algorithm created here. The problem with the batching algorithm that is used in the current solution is that it is a heuristic one; this means that while the solutions are better than the ones formed by the company algorithms, they are still not optimal. The standard approach to finding optimal batching solutions is using LP models. However, since the batching in this company should be recalculated every few minutes, the time it takes to solve the LP model is unrealistic for the intended use. That is why we recommend that the company use genetic algorithms to find better solutions, mainly because of the faster runtimes, which is more appropriate for the current problem.

Lastly, if the company wants to improve their warehousing operations even further, the following steps would be to optimise the storage assignment of the warehouse as well as introduce a more synchronised zoning system into its warehouses. The company's current storage assignment method is a dedicated storage policy. This policy is one of the most common policies used in warehousing due to its simplicity of implementation. However, the main drawback of this allocation policy is that it does not continually optimise for shorter route constructions. That is why we recommend that the company changes to a full turnover or class-based storage policies, which, according to Koster, De-Luc and Roodbergen (2006), provide much shorter routes and better batches. Since we know that the company cannot stop its operations for a few days to reorganise the warehouse due to the high amount of lost potential sales, we recommend gradually implementing such a policy. The gradual implementation would work by creating an ideal warehouse layout by using, for example, a

class-based storage policy. Then, as one product gets out of stock, a product that would be in an ideal layout would be placed into the location of the product that just went out of stock. By swapping these products, the layout would gradually change into an ideal one within a few months. As for the zoning systems, the warehouses currently consist of two zones, the bottled and the FMCGs. However, some warehouses are topologically divided into even more sections, which could naturally be turned into their respective zones. For example, the company warehouse in Münster has its FMCG zone divided into three rooms: food, frozen goods, and household items. In this scenario, dividing the FMCG zone into three smaller ones could benefit the company. It is important to note that adding additional zones does not always improve performance and needs to be evaluated for each case. We hope that the ideas listed here benefit the company and that by implementing them, they increase the performance of the warehousing operations within Flaschenpost.

Bibliography

Borkar, V. S., Ejov, V., Filar, J. A., & Nguyen, G. T. (2012). *Hamiltonian Cycle Problem and Markov Chains.* New York: Springer.

Bozer, Y. A., & Kile, J. W. (2008). Order batching in walk-and-pick order picking systems. *International Journal of Production Research*, 1887–1909.

Briant, O., Cambazard, H., Cattaruzza, D., Catusse, N., Ladier, A.-L., & Ogier, M. (2020). An efficient and general approach for the joint order batching and picker routing problem. *European Journal of Operational Research*, 497-512.

Cergibozan, Ç., & Tasan, A. S. (2016). Order batching operations: an overview of classification, solution techniques, and future research. *Journal of Intelligent Manufacturing*, 335–349.

Clarke, G., & Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 568-581.

Cooper, D., & Schindler, P. (2014). *Business research methods.* McGraw-Hill. Retrieved from https://thuvienso.hoasen.edu.vn/handle/123456789/10310

Cornuéjols, G., Fonlupt, J., & Naddef, D. (1985). The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical programming*, 1-27.

ecommerceDB.com. (2023, July). *Top online stores in the Food & Beverages segment in Germany in 2022, by e-commerce net sales*. Retrieved from Statista: https://www.statista.com/forecasts/871179/top-online-stores-food-beverages-germany-ecommercedb

Gademann, N., & Velde, S. v. (2005). Order batching to minimize total travel time in a parallel-aisle warehouse. *IIE Transactions*, 62 - 75.

GfK. (2024, January). *Revenue in food retail in Germany from 2020 to 2023*. Retrieved from Statista: https://www.statista.com/statistics/1423454/food-retail-revenue-germany/#:~:text=In%202023%2C%20German%20food%20retail,two%20years%20in%20the%20timeline

GfK. (2024, January). *Year-on- year revenue development of FMCG products in food retail in Germany in 2023, by category*. Retrieved from Statista: https://www.statista.com/statistics/857704/revenue-development-fmcg-selection-food-retail-germany/

Hans Heerkens, A. v. (2017). *Solving Managerial Problems Systematically.* Groningen: Noordhoff Uitgevers.

IfD Allensbach. (2023, June). *Number of people in Germany who purchase groceries for their household on the internet or from online shops from 2019 to 2023*. Retrieved from Statista: https://www.statista.com/statistics/989726/purchasing-groceries-on-the-internet-or-from-online-shops/

Il-Choe, K., & Sharp, G. (1991). *SMALL PARTS ORDER PICKING: DESIGN AND OPERATION.* Atlanta: Georgia Tech Research Corporation.

Islam, M. S., & Uddin, M. K. (2023). Correlated Storage Assignment Approach in Warehouses: A Systematic Literature Review. *Journal of Industrial Engineering and Management*, 294 -318.

Jungnickel, D. (2013). The Greedy Algorithm. In D. Jungnickel, *Graphs, Networks and Algorithms* (pp. 135-163). Berlin: Springer.

Kapoor, A. P., & Vij, M. (2021). Following you wherever you go: Mobile shopping 'cart-checkout' abandonment. *Journal of Retailing and Consumer Services, 61*. doi:https://doi.org/10.1016/j.jretconser.2021.102553

Koster, M. B., Poort, E. S., & Wolters, M. (1999). Efficient orderbatching methods in warehouses. *International Journal of Production Research*, 1479-1504.

Koster, R. d., Le-Duc, T., & Roodbergen, K. J. (2006). Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 481 - 498.

Kulak, O., Sahin, Y., & Taner, M. E. (2011). Joint order batching and picker routing in single and multiple-cross-aisle warehouses using cluster-based tabu search algorithms. *Flexible Services and Manufacturing Journal*, 52-80.

Kusiak, A., Vannelli, A., & Kumar, K. R. (1986). Clustering analysis: models and algorithms. *Control Cyberne*, 139-154.

Martello, S., & Toth, P. (1987). Algorithms for Knapsack Problems. *North-Holland Mathematics Studies*, 123-257.

Murtagh, E., Mair, J., Aguiar, E., Tudor-Locke, C., & Murphy, M. (2021). Outdoor Walking Speeds of Apparently Healthy Adults: A Systematic Review and Meta-analysis. *Sports Med*, 125-141.

Muter, İ., & Öncan, T. (2015). An exact solution approach for the order batching. *IIE Transactions*, 728 - 738.

Ratlif, H. D., & Rosenthal, A. S. (1982). Order-Picking in a Rectangular Warehouse: A Solvable Case of the Traveling Salesman Problem. *Open Journal of Business and Management*.

Roodbergen, K. J., & Koster, R. D. (2001). Routing order pickers in a warehouse with a middle aisle. *European Journal of Operational Research*, 32-43.

Vallea, C. A., Beasley, J. E., & Cunha, A. S. (2017). Optimally solving the joint order batching and picker routing problem. *European Journal of Operational Research*, 817 - 834.

Vinod, H. D. (1969). Integer Programming and the Theory of Grouping. *Journal of the American Statistical Association*, 506–519.

Appendix

Appendix 1 (Research Design)

The following *Table 1* shows the overview of the research design that will be used throughout the thesis.

Research question	**How to reduce average order picking time in the warehouse picking process of Flaschenpost?**						
Knowledge questions	**MPSM phase**	**Research population**	**Research type**	**Data gathering method**	**Research strategy**	**Research outcome**	**Action plan**
What are the current methods and models being used in the company for the picking process?	3	Company employees & supervisor	Descriptive	Unstructured interview & primary sources[12]	Qualitative	Overview of the current picking process method	-Interview -Read about the company -Write an overview
What models and heuristics currently exist that deal with the warehouse picking problem?	3 & 4	Literature & Databases	Descriptive	Literature study	Qualitative	List of possible solutions to the problem	-Read research papers -Analyse them -Consult with the mentor
Which KPIs are relevant for the improvement of the Flaschenpost' s order throughput rate?	5	Company supervisor & Literature theory	Descriptive	Unstructured interviews & Literature study	Quantitative	List of KPIs that I will use to select the best solution	-Read research papers -Consult the company supervisor -Make a list of KPIs
How can these algorithms that exists be modified such that they are applicable to our picking process?	6	Company datasets, Literature, Company supervisor	Explanatory	Companies' database data & Literature study	Qualitative	Mathematical model and a heuristic	-Select a solution -formulate the constraints -formulate the objective function
How to evaluate the mathematical models and heuristics and its impact on the Flaschenpost' s warehouse picking process?	7	Company order datasets, Literature, Databases	Explanatory	Primary sources & Literature study	Quantitative	A detailed analysis of the model and verifying its performance	-Read the research papers -make a simulation -run the simulation

[12] Primary sources - annual reports and other internal company data

What are the next steps the company should take and future recommendations on how to further improvement of our solution?	7	Literature theory & Company supervisor	Descriptive	Unstructured interviews & Literature study	Qualitative	A list of recommendations and future improvements	-Summarize the knowledge obtained throughout the thesis -write a detailed plan on how to improve the solution

Appendix 2 (S-Shaped heuristic)

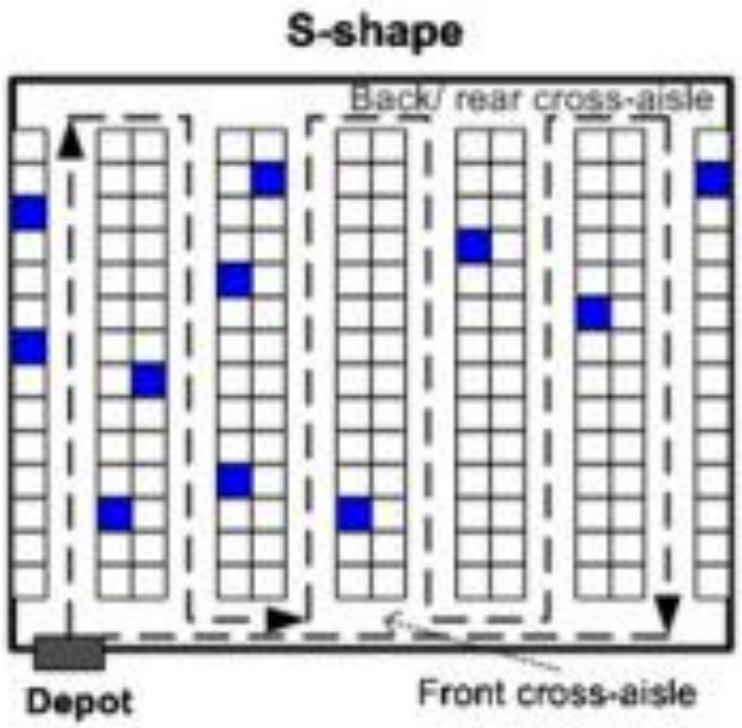

Figure 22: An example of a route using an S-shape heuristic (Koster, De-Luc, Roodbergen, 2001)

Appendix 3 (State transition tables)

L_j^- to L_j^{+y}	(1)	(2)	(3)	(4)	(5)	(6)[a]
$(0,0,0,0)$[b]	$(0,u,u,1)$	$(0,e,e,1)$	$(0,e,0,1)$	$(0,0,e,1)$	$(0,e,e,2)$	$(0,0,0,0)$
$(0,0,0,1)$[c]	[d]	[d]	[d]	[d]	[d]	$(0,0,0,1)$
$(e,0,0,1)$	$(e,u,u,2)$	$(e,e,e,2,a–bc)$	$(e,e,0,2)$	$(e,0,e,2)$	$(e,e,e,3)$	$(e,0,0,1)$
$(0,e,0,1)$	$(0,u,u,1)$	$(0,e,e,1)$	$(0,e,0,1)$	$(0,e,e,2)$	$(0,e,e,2)$	$(0,e,0,1)$
$(0,0,e,1)$	$(0,u,u,1)$	$(0,e,e,1)$	$(0,e,e,2)$	$(0,0,e,1)$	$(0,e,e,2)$	$(0,0,e,1)$
$(e,e,0,1)$	$(e,u,u,1)$	$(e,e,e,1)$	$(e,e,0,1)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$	$(e,e,0,1)$
$(e,0,e,1)$	$(e,u,u,1)$	$(e,e,e,1)$	$(e,e,e,2,b–ac)$	$(e,0,e,1)$	$(e,e,e,2,b–ac)$	$(e,0,e,1)$
$(0,e,e,1)$	$(0,u,u,1)$	[e]	$(0,e,e,1)$	$(0,e,e,1)$	$(0,e,e,1)$	$(0,e,e,1)$
$(e,e,e,1)$	$(e,u,u,1)$	[e]	$(e,e,e,1)$	$(e,e,e,1)$	$(e,e,e,1)$	$(e,e,e,1)$
$(u,u,0,1)$	$(u,e,u,1)$	$(u,u,e,1)$	$(u,u,0,1)$	$(u,u,e,2)$	$(u,u,e,2)$	$(u,u,0,1)$
$(u,0,u,1)$	$(u,u,e,1)$	$(u,e,u,1)$	$(u,e,u,2)$	$(u,0,u,1)$	$(u,e,u,2)$	$(u,0,u,1)$
$(0,u,u,1)$	$(0,e,e,1)$	[e]	$(0,u,u,1)$	$(0,u,u,1)$	$(0,u,u,1)$	$(0,u,u,1)$
$(e,u,u,1)$	$(e,e,e,1)$	[e]	$(e,u,u,1)$	$(e,u,u,1)$	$(e,u,u,1)$	$(e,u,u,1)$
$(u,e,u,1)$	$(u,u,e,1)$	[e]	$(u,e,u,1)$	$(u,e,u,1)$	$(u,e,u,1)$	$(u,e,u,1)$
$(u,u,e,1)$	$(u,e,u,1)$	[e]	$(u,u,e,1)$	$(u,u,e,1)$	$(u,u,e,1)$	$(u,u,e,1)$
$(e,e,0,2)$	$(e,u,u,2)$	$(e,e,e,2,a–bc)$	$(e,e,0,2)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,e,0,2)$
$(e,0,e,2)$	$(e,u,u,2)$	$(e,e,e,2,a–bc)$	$(e,e,e,3)$	$(e,0,e,2)$	$(e,e,e,3)$	$(e,0,e,2)$
$(0,e,e,2)$	$(0,u,u,1)$	$(0,e,e,1)$	$(0,e,e,2)$	$(0,e,e,2)$	$(0,e,e,2)$	$(0,e,e,2)$
$(e,e,e,2,a–bc)$	$(e,u,u,2)$	[e]	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$
$(e,e,e,2,b–ac)$	$(e,u,u,1)$	$(e,e,e,1)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$
$(e,e,e,2,c–ab)$	$(e,u,u,1)$	$(e,e,e,1)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$
$(e,u,u,2)$	$(e,e,e,2,a–bc)$	[e]	$(e,u,u,2)$	$(e,u,u,2)$	$(e,u,u,2)$	$(e,u,u,2)$
$(u,e,u,2)$	$(u,u,e,1)$	$(u,e,u,1)$	$(u,e,u,2)$	$(u,e,u,2)$	$(u,e,u,2)$	$(u,e,u,2)$
$(u,u,e,2)$	$(u,e,u,1)$	$(u,u,e,1)$	$(u,u,e,2)$	$(u,u,e,2)$	$(u,u,e,2)$	$(u,u,e,2)$
$(e,e,e,3)$	$(e,u,u,2)$	$(e,e,e,2,a–bc)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,e,e,3)$

[a] This transition is only allowed if there are no items in this part of the aisle.
[b] This class can occur only if there are no items to be picked in L_j^-.
[c] This class can only occur if there are no items to be picked in $G - L_j^-$.
[d] This transition would violate condition A2(c).
[e] This transition will never lead to the optimal solution.

Figure 23: Transition state table from stage L_j^- to L_j^{+y} (Koster, Roodbergen, 2001)

L_j^{+y} to L_j^{+x}	(1)	(2)	(3)	(4)	(5)	(6)[a]
$(0,0,0,0)$[b]	$(u,u,0,1)$	$(e,e,0,1)$	$(e,0,0,1)$	$(0,e,0,1)$	$(e,e,0,2)$	$(0,0,0,0)$
$(0,0,0,1)$[c]	[d]	[d]	[d]	[d]	[d]	$(0,0,0,1)$
$(e,0,0,1)$	$(u,u,0,1)$	$(e,e,0,1)$	$(e,0,0,1)$	$(e,e,0,2)$	$(e,e,0,2)$	$(e,0,0,1)$
$(0,e,0,1)$	$(u,u,0,1)$	$(e,e,0,1)$	$(e,e,0,2)$	$(0,e,0,1)$	$(e,e,0,2)$	$(0,e,0,1)$
$(0,0,e,1)$	$(u,u,e,2)$	$(e,e,e,2,c–ab)$	$(e,0,e,2)$	$(0,e,e,2)$	$(e,e,e,3)$	$(0,0,e,1)$
$(e,e,0,1)$	$(u,u,0,1)$	[e]	$(e,e,0,1)$	$(e,e,0,1)$	$(e,e,0,1)$	$(e,e,0,1)$
$(e,0,e,1)$	$(u,u,e,1)$	$(e,e,e,1)$	$(e,0,e,1)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$	$(e,0,e,1)$
$(0,e,e,1)$	$(u,u,e,1)$	$(e,e,e,1)$	$(e,e,e,2,a–bc)$	$(0,e,e,1)$	$(e,e,e,2,a–bc)$	$(0,e,e,1)$
$(e,e,e,1)$	$(u,u,e,1)$	[e]	$(e,e,e,1)$	$(e,e,e,1)$	$(e,e,e,1)$	$(e,e,e,1)$
$(u,u,0,1)$	$(e,e,0,1)$	[e]	$(u,u,0,1)$	$(u,u,0,1)$	$(u,u,0,1)$	$(u,u,0,1)$
$(u,0,u,1)$	$(e,u,u,1)$	$(u,e,u,1)$	$(u,0,u,1)$	$(u,e,u,2)$	$(u,e,u,2)$	$(u,0,u,1)$
$(0,u,u,1)$	$(u,e,u,1)$	$(e,u,u,1)$	$(e,u,u,2)$	$(0,u,u,1)$	$(e,u,u,2)$	$(0,u,u,1)$
$(e,u,u,1)$	$(u,e,u,1)$	[e]	$(e,u,u,1)$	$(e,u,u,1)$	$(e,u,u,1)$	$(e,u,u,1)$
$(u,e,u,1)$	$(e,u,u,1)$	[e]	$(u,e,u,1)$	$(u,e,u,1)$	$(u,e,u,1)$	$(u,e,u,1)$
$(u,u,e,1)$	$(e,e,e,1)$	[e]	$(u,u,e,1)$	$(u,u,e,1)$	$(u,u,e,1)$	$(u,u,e,1)$
$(e,e,0,2)$	$(u,u,0,1)$	$(e,e,0,1)$	$(e,e,0,2)$	$(e,e,0,2)$	$(e,e,0,2)$	$(e,e,0,2)$
$(e,0,e,2)$	$(u,u,e,2)$	$(e,e,e,2,c–ab)$	$(e,0,e,2)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,0,e,2)$
$(0,e,e,2)$	$(u,u,e,2)$	$(e,e,e,2,c–ab)$	$(e,e,e,3)$	$(0,e,e,2)$	$(e,e,e,3)$	$(0,e,e,2)$
$(e,e,e,2,a–bc)$	$(u,u,e,1)$	$(e,e,e,1)$	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$	$(e,e,e,2,a–bc)$
$(e,e,e,2,b–ac)$	$(u,u,e,1)$	$(e,e,e,1)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$	$(e,e,e,2,b–ac)$
$(e,e,e,2,c–ab)$	$(u,u,e,2)$	[e]	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$	$(e,e,e,2,c–ab)$
$(e,u,u,2)$	$(u,e,u,1)$	$(e,u,u,1)$	$(e,u,u,2)$	$(e,u,u,2)$	$(e,u,u,2)$	$(e,u,u,2)$
$(u,e,u,2)$	$(e,u,u,1)$	$(u,e,u,1)$	$(u,e,u,2)$	$(u,e,u,2)$	$(u,e,u,2)$	$(u,e,u,2)$
$(u,u,e,2)$	$(e,e,e,2,c–ab)$	[e]	$(u,u,e,2)$	$(u,u,e,2)$	$(u,u,e,2)$	$(u,u,e,2)$
$(e,e,e,3)$	$(u,u,e,2)$	$(e,e,e,2,c–ab)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,e,e,3)$	$(e,e,e,3)$

[a] This transition is only allowed if there are no items in this part of the aisle.
[b] This class can occur only if there are no items to be picked in L_j^{+y}.
[c] This class can only occur if there are no items to be picked in $G - L_j^{+y}$.
[d] This transition would violate condition A2(c).
[e] This transition will never lead to the optimal solution.

Figure 24: Transition state table from stage L_j^{+y} to L_j^{+x} (Koster, Roodbergen, 2001)

L_j^{+x} to L_{j+1}^{-}[a]	(1)	(2)	(3)	(7)	(8)	(9)		
$(u,u,0,1)$	$(u,u,0,1)$	[b]	[b]	[b]	[b]	[c]		
$(u,0,u,1)$	[b]	$(u,0,u,1)$	[b]	[b]	[c]	[b]		
$(0,u,u,1)$	[b]	[b]	$(0,u,u,1)$	[c]	[b]	[b]		
$(e,u,u,1)$	[b]	[b]	$(0,u,u,1)$	$(e,u,u,1)$	[b]	[b]		
$(u,e,u,1)$	[b]	$(u,0,u,1)$	[b]	[b]	$(u,e,u,1)$	[b]		
$(u,u,e,1)$	$(u,u,0,1)$	[b]	[b]	[b]	[b]	$(u,u,e,1)$		
$(e,u,u,2)$	[b]	[b]	[d]	$(e,u,u,2)$	[b]	[b]		
$(u,e,u,2)$	[b]	[d]	[b]	[b]	$(u,e,u,2)$	[b]		
$(u,u,e,2)$	[d]	[b]	[b]	[b]	[b]	$(u,u,e,2)$		
L_j^{+x} to L_{j+1}^{-}	(4)	(5)	(6)	(10)	(11)	(12)	(13)	(14)
$(0,0,0,0)$[e]	[e]	[e]	[e]	[e]	[e]	[e]	[e]	$(0,0,0,0)$
$(0,0,0,1)$[f]	[d]	[d]	[d]	[d]	[d]	[d]	[d]	$(0,0,0,1)$[g]
$(e,0,0,1)$	$(e,0,0,1)$	[d]	[d]	[c]	[c]	[d]	[c]	$(0,0,0,1)$[g]
$(0,e,0,1)$	[d]	$(0,e,0,1)$	[d]	[c]	[d]	[c]	[c]	$(0,0,0,1)$[g]
$(0,0,e,1)$	[d]	[d]	$(0,0,e,1)$	[d]	[c]	[c]	[c]	$(0,0,0,1)$[g]
$(e,e,0,1)$	$(e,0,0,1)$	$(0,e,0,1)$	[d]	$(e,e,0,1)$	[c]	[c]	[c]	$(0,0,0,1)$[g]
$(e,0,e,1)$	$(e,0,0,1)$	[d]	$(0,0,e,1)$	[c]	$(e,0,e,1)$	[c]	[c]	$(0,0,0,1)$[g]
$(0,e,e,1)$	[d]	$(0,e,0,1)$	$(0,0,e,1)$	[c]	[c]	$(0,e,e,1)$	[c]	$(0,0,0,1)$[g]
$(e,e,e,1)$	$(e,0,0,1)$	$(0,e,0,1)$	$(0,0,e,1)$	$(e,e,0,1)$	$(e,0,e,1)$	$(0,e,e,1)$	$(e,e,e,1)$	$(0,0,0,1)$[g]
$(e,e,0,2)$	[d]	[d]	[d]	$(e,e,0,2)$	[d]	[d]	[e]	[d]
$(e,0,e,2)$	[d]	[d]	[d]	[d]	$(e,0,e,2)$	[d]	[e]	[d]
$(0,e,e,2)$	[d]	[d]	[d]	[d]	[d]	$(0,e,e,2)$	[e]	[d]
$(e,e,e,2,a–bc)$	[d]	[d]	[d]	$(e,e,0,2)$	$(e,0,e,2)$	[d]	$(e,e,e,2,a–bc)$	[d]
$(e,e,e,2,b–ac)$	[d]	[d]	[d]	$(e,e,0,2)$	[d]	$(0,e,e,2)$	$(e,e,e,2,b–ac)$	[d]
$(e,e,e,2,c–ab)$	[d]	[d]	[d]	[d]	$(e,0,e,2)$	$(0,e,e,2)$	$(e,e,e,2,c–ab)$	[d]
$(e,e,e,3)$	[d]	[d]	[d]	[d]	[d]	[d]	$(e,e,e,3)$	[d]

[a] All combinations not in this table would violate condition A2(b).
[b] This transition would violate condition A2(b).
[c] This transition will never lead to the optimal solution.
[d] This transition would violate condition A2(c).
[e] This class can occur only if there are no items to be picked in L_j^{+x}.
[f] This class can only occur if there are no items to be picked in $G - L_j^{+x}$.
[g] This transition is only allowed if there are no items to be picked in $G - L_j^{+x}$.

Figure 25: Transition state table from stage L_j^{+x} to L_j^{-} (Koster, Roodbergen, 2001)

UNIVERSITY OF TWENTE
Drienerlolaan 5
7522 NB Enschede

P.O.Box 217
7500 AE Enschede

P +31 (0)53 489 9111

info@utwente.nl www.utwente.nl