AN ALGORITHM FOR THE BAY PLACEMENT OF INTERMODAL CONTAINERS

L. R. van Beek
Industrial Engineering & Management (s1805665)
University of Twente

AN ALGORITHM FOR THE BAY PLACEMENT OF INTERMODAL CONTAINERS

Bachelor Thesis

Enschede, September 2019

Student
L.R. van Beek (Leon)
Industrial Engineering and Management (s1805665)
University of Twente

Lead supervisor
Dr. Ir. E.A. Lalla (Eduardo)
Assistant Professor
University of Twente

Second supervisor
Dr. Ir. M.R.K. Mes (Martijn)
Associate Professor
University of Twente

Supervisor Cofano B.V.
L. de Vries
Lead Software Engineer
Cofano Software Solutions

EXECUTIVE SUMMARY

INTRODUCTION

Cofano Software Solutions is a software development company that develops applications for the logistics industry. Their focus is the container freight transportation industry. Cofano creates and adapts applications to the wishes of their customers in order to improve performance in various aspects. The offered bachelor assignment is about optimizing the way containers are stacked at a terminal such that the amount of internal relocation moves decrease.

Currently, the containers at the terminal are stacked in a non-optimal manner without any optimization method. As relocation moves cost time and money, one aims to reduce these. This can be done by implementing an automated stacking method that places the containers at the terminal. Therefore, in this research the following core problem is solved: *"There is no computed method behind the current policy of container stacking."*. The solving of the core problem is done by designing an algorithm that places the containers at the terminal, and an evaluation of the results to check whether actual improvement is made compared to the current system.

APPROACH

In order to be able to construct the algorithm, a literature review is conducted, a data analysis performed and expertise knowledge from the company and UT supervisors considered.

In the literature review the following was found:

- The problem addressed in this research belongs to the class of Loading and Unloading (LU) problems.
- An *event-based* approach will be used.
- Examples of models / heuristics previously used, that can give insights into pitfalls and possible solutions.

From the data analysis the following was found:

- It is possible to give containers an estimated leave date per carrier based on the historical data.
- The data provides the necessary information to make the results from the algorithm comparable to the real life situation.
- The data gives the background knowledge needed on the workings of a container terminal, which helps the development the algorithm.

In the current stacking policy, the containers are stacked according to their corresponding carrier. All containers are put together in clumps of the same carrier, without paying any attention to their estimated departure dates (Etd's). The goal is to develop an algorithm that does consider the Etd's of containers and stack them according to these. Challenges faced are the determination on how to improve the method of stacking, coping with the restrictions posed by the reach stacker, and ensuring the validity op the proposed stacking method.

THE ALGORITHM

The algorithm that is devised places the containers at the terminal according to their Estimated departure date (Etd). Containers are firstly placed on top of containers with the same Etd, secondly on

a ground level, and thirdly on top of the container with the highest Etd. With all stacking being performed under stacking restrictions posed by a reach stacker (container transportation vehicle). The algorithm is implemented into Visual Basic for Applications (VBA). In order to examine the validity of the algorithm and to make the results comparable, several other stacking methods have also been implemented into VBA. The three stacking methods that have been implemented into VBA are:

- A random stacking method
- The current stacking method
- The new stacking method

To give further options for improvement and to show how the new stacking method performs when stacking containers with a yard crane the possible implementation of a yard crane has also been examined. The assessment of the yard crane performance also broadens the possibilities for implementation at larger terminals, as larger terminals already often have a yard crane system. Thus, the following two stacking methods have also been implemented into VBA:

- The implementation of a yard crane using the current stacking method.
- The implementation of a yard crane using the new stacking method.

These five different methods are tested with three databases of different sizes. All the inputs for the stacking methods were the same to ensure the validity of the results. From the results can be seen that the new stacking method significantly decreases the relocation percentage compared to the current stacking method (the simulated one, as well as the real life one). As well as on the random stacking method. For the implementation of a yard crane two test instances are used. The stacking with a yard crane using the current stacking method and the stacking of a yard crane using the new stacking method. The implementation of a yard crane using the current stacking method outperforms the current stacking method using a reach stacker, but not the new stacking method with the use of a reach stacker. The yard crane implementation stacking with the new stacking method outperforms all other stacking methods. This was to be expected as a yard crane has fewer stacking restrictions compared to a reach stacker and uses the newly devised stacking method.

Assumptions

In order to be able to develop an algorithm the following assumptions are made:

- Containers can only be (re)stacked in one bay.
- The estimated arrival or departure date of the container is also the actual arrival or departure date in the algorithm. Which can be assumed because this is the case for the larger part of the containers.
- All containers have a known Etd.
- Containers that need to be placed in areas designated for dangerous goods are not processed by the algorithm.
- Truck retrieval and return happens on one day.
- Containers of different sizes get stacked in different bays.
- All the containers that arrive on a day are placed before retrieval starts.

The assumptions mentioned above made it possible to develop the algorithm. The main issue in which the algorithm differs from the real life situation is the case of the known or unknown Etd's. In order

for the algorithm to work all containers must receive an Etd. This is not yet the case in the real life situation. However, once this is fixed the algorithm can be used in practice. In this research a possible way of giving Etd's to containers is proposed. Containers can receive an Etd based on the average time containers of corresponding carrier stay at the terminal. These average times are calculated by analysing the dataset provided by Cofano and can be found in Figure i. To ensure data confidentiality the names of the carriers have been replaced with numbers.

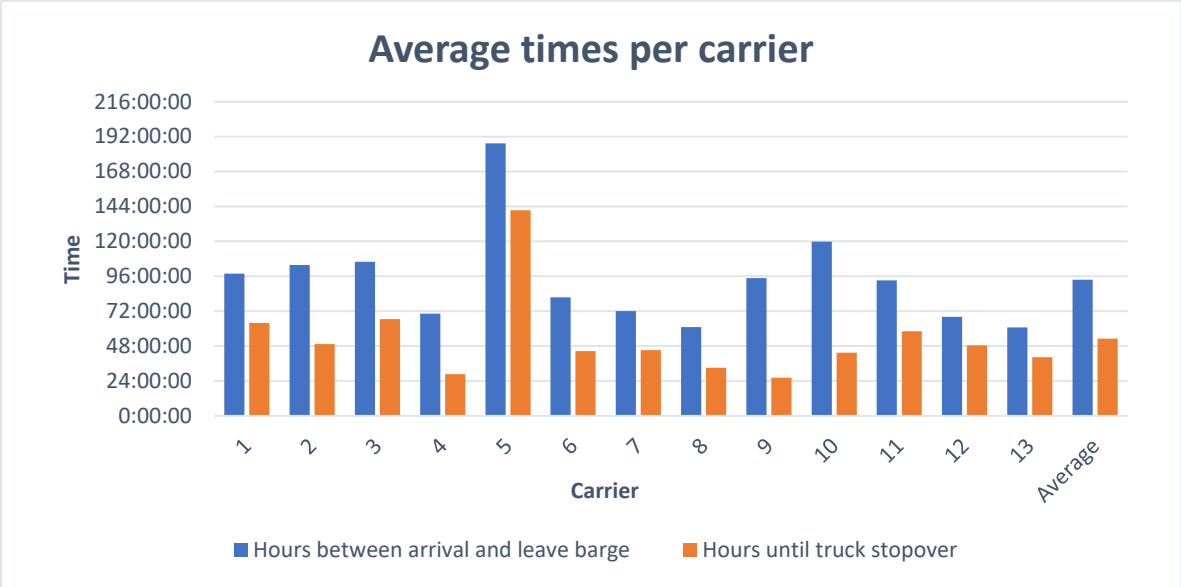

Figure i: Average times per carrier (anonymised)

CONCLUSIONS AND RECOMMENDATIONS

With the outcomes of the research we have shown that the new stacking method outperforms the current one under the assumptions of the model. The assumptions which made it possible to construct the algorithm are explained and their impact on the performance of the algorithm analysed. Once all containers receive an Etd the algorithm is ready to be used in practice and reduction in relocation moves can be expected as the improvement in the simulated environment is significant.

Before the implementation of the algorithm, Cofano should look whether they can improve the algorithm exploring the options of assigning Etd's to all arriving containers. The proposed algorithm improves on the current stacking method reducing the number of relocation moves. The algorithm is however not yet ready to be implemented into an application that places containers because all containers first need to receive an Etd before the algorithm is capable of placing them at the terminal.

From the research the following recommendations for Cofano follow:

- Find a way to assign all containers an Etd in order to be able to implement the proposed algorithm.
- Test the proposed algorithm on the data of multiple terminals to ensure that performance is similar when implementing the algorithm into an application.
- Explore options for improvement of the algorithm mentioned in the *"recommendations for future research"* subsection in Chapter 6.

PREFACE

This research contains the bachelor's thesis *"An algorithm for the bay placement of intermodal containers".* This thesis was written in order to complete my bachelor Industrial Engineering and Management at the University of Twente. The research has been conducted at Cofano Software Solutions considering the gathered data of a customer of theirs. The research proposes a new stacking method with the goal of improving container handling operations at terminals. This research has been conducted from May the 27th, 2019 till the 4th of October, 2019.

I would like to thank my supervisor Leon de Vries from Cofano and all other employees for their support, feedback, and willingness to answer my questions. Furthermore, I would like to thank my supervisors E.A. Lalla and M.R.K. Mes from the University of Twente. They were available for timely meetings and their insight, experience and feedback helped greatly in conducting this research.

Leon van Beek

Enschede, 27 September

TABLE OF CONTENTS

EXECUTIVE SUMMARY ..... **III**

PREFACE ..... **VI**

LIST OF DEFINITIONS ..... **VIII**

1. INTRODUCTION **1**

1.1 Current situation *1*
1.2 Problem Description *2*
1.3 Problem approach and available material *3*
1.4 Research questions *4*
1.5 Context terminal *5*
1.6 Deliverables *8*

2. LITERATURE REVIEW **9**

2.1 Related works *9*
2.2 Classification of literature to research *13*
2.3 Conclusion *14*

3. SOLUTION APPROACH **15**

3.1 Data gathering *15*
3.2 Data analysis *16*
3.3 Solution design *21*
3.4 Conclusion *23*

4. NEW STACKING METHOD **25**

4.1 Algorithm *25*
4.2 Implementation *30*
4.3 Conclusion *37*

5. EVALUATION OF RESULTS **38**

5.1 Experiment results *38*
5.2 quality of the generated solution *43*
5.3 conclusion *43*

6. CONCLUSIONS, RECOMMENDATIONS AND LIMITATIONS **45**

6.1 Conclusions *45*
6.2 Recommendations *46*
6.3 Limitations *47*

REFERENCES **49**

APPENDIX A: ENTIRE PSEUDOCODE NEW STACKING METHOD **50**

APPENDIX B: VISUAL REPRESENTATION YARD BAY **53**

APPENDIX C: COMPLETE RESULTS **54**

LIST OF DEFINITIONS

Intermodal Shipping Container	Large standardized shipping container designed and built for intermodal freight transport. Used to transports good by means of barge, train and truck.
Barge	Containership used for transporting intermodal containers over inland waterways.
Bay	A ground “matrix” where containers can be placed.
Heuristic	A heuristic is a method for finding a solution to a problem that employs a practical method that does not providing the optimal solution, but a reasonably good solution. Heuristics are generally used as they require limited computation times.
Relocation Move	A relocation move occurs in a terminal when a container needs to be moved internally in order to get to the underlying container which needs to be moved to their designated means of transportation.
Remarshalling	Relocating containers without the need to pick up a container underneath it, to prevent the future relocation of containers.
Blocks Relocation Problem (BRP)	The problem of which blocks to move in order to minimize the total number of moves needed to retrieve containers from a terminal.
Reach Stacker	A reach stacker is a transportation devise that hoists containers. It is used for loading and unloading containers in small or medium-sized terminals.
Stacking Problem (SP)	An optimization problem in which the flow of blocks (i.e. containers) in storage is optimized. The objective of an SP solution is to minimize the number of movements required to store the blocks.

1. INTRODUCTION

In this part, the problems that arise in container stacking will be introduced. Firstly, the current situation will be explained in Section 1.1. After that the core problem will be determined in Section 1.2. With this core problem, certain research questions arise that need to be answered in order to solve the core problem. These will be discussed in Section 1.3.

1.1 CURRENT SITUATION

The bachelor assignment is at Cofano, a software development company. They develop logistical applications with some of their customers being Schiphol and the port of Rotterdam. The establishment located in Enschede focusses on the development of logistical applications that their customers can use to, for example, track their intermodal shipping containers. The assignment itself is about finding a method of container stacking that outperforms other current stacking methods at terminals. The aim of Cofano is to have a new stacking method reducing relocation moves that can be implemented at multiple terminals, but in this research the historical data of a single terminal is used. The containers at the terminal are stored for some time until further transportation. The transportation is currently being conducted by ship and truck, and in the future possibly by train. The offered assignment is about optimizing the way these containers are stacked such that the amount of internal relocation moves, and possibly transportation distances are decreased. This research will be set up by composing a solution approach that determines where to place the intermodal containers for storage in a terminal. The terminal that is used for validation is a relatively small inland terminal with a capacity of about 2000 containers. The name cannot be mentioned because of data confidentiality constraints. This terminal has been selected as the dataset that Cofano has gathered is most complete at this specific terminal. The terminal is however not owned by Cofano, as they are a software development company. So, the research will be conducted at Cofano, considering the terminal of a single customer of theirs. Cofano wants to develop a software application that determines where to place the containers at the terminal, and then lease or sell the software to them. The goal is to have a solution that can also be easily adapted and implemented at other terminals.

The problem that arises at the terminal is that there is no computed method of container stacking with the current stacking policy causing unnecessary relocation moves that are costly and could be avoided. In the current policy, the empty containers when placed at the terminal are put together in clumps considering the carrier of the containers. When the containers are full, they are placed according to their designated export modality (e.g. all containers that need to leave by truck are stacked together). This however, results in more relocation moves than necessary, as there is no consideration for the estimated departure date. It is not taken into account when the containers need to leave again, and in this aspect, improvements can be made. Furthermore, empty containers are stacked onto each other and are interchangeable when being handled by the same carrier and are of the same type.

Taking into account the transportation distance problem, the port often needs to move the containers further than necessary, they pay no attention to where the containers need to leave when placing them at the terminal. Because of this the containers often need to be transported a longer distance than necessary. Figure 1 below offers some more explanation considering the transportation within, and layout of, the terminal.

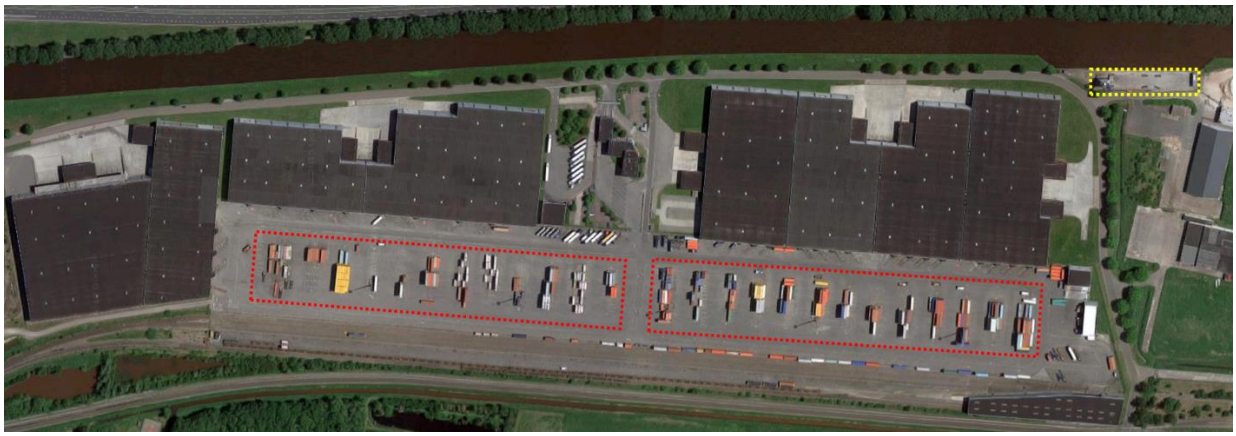

Figure 1: Terminal Layout **Source:** adapted from Google Maps[1].

Figure 1 shows the layout of the terminal. The yellow area depicts the loading / unloading dock for barges, and red areas represent bays where containers can be stacked for storage. When containers arrive by truck or barge, they are stored at the terminal until further transportation. From the Google maps image, the yard layout can be derived. In the yard five containers can be stacked next to each other per bay, and the length per bay is either six 20 FT containers, four 30 Feet or three 40 FT containers. The containers can be stacked 6 high, resulting in the following respective dimensions (L x W x H). (6 x 5 x 6), (4 x 5 x 6) and (3 x 5 x 6). In total, there are 25 bays in which containers can be stored. All containers are transported by means of reach stackers.

1.2 PROBLEM DESCRIPTION

In Figure 2 the problem cluster with the flow of problems at the terminal is displayed. This diagram helps visualise the causes of the problems at the terminal and is useful in helping determine the core problem.

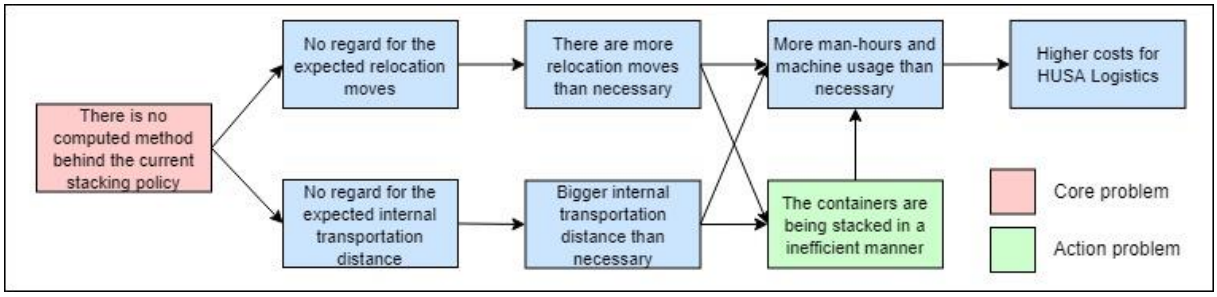

Figure 2: Problem cluster displaying the flow of problems at the terminal.

From Figure 2 it can be seen that at the terminal the containers are placed together such that the employees operating the machines will find them easy to identify. As the containers are placed in this manner, there is no consideration for the internal transportation distance and the amount of relocation moves that are necessary. That these two factors are not taken into account results in an inefficient stacking method, and more man-hours and machine usage than necessary, resulting in higher costs. Once the terminal takes these factors in consideration cost reduction can be achieved.

[1] https://www.google.com/maps/

Thus, Cofano wants to solve this problem for terminal, as they themselves do not pay attention to this inefficiency. Once the core problem is solved, the action problem itself will also disappear.

Core problem

Considering the problem cluster shown in Figure 2, the core problem is determined. The cause for the problems at the terminal is that they have not yet thought of a stacking method that incorporates important factors that influence the efficiency of it. Thus, the core problem is defined as follows:

"There is no computed method behind the current policy of container stacking."

Following the problem cluster there are two factors that influence this inefficient stacking method and can thus contribute to solving the action problem. These two factors are:

"the amount of unnecessary relocation moves at the terminal." and

"The big internal transportation distance at the terminal."

As there are currently a lot of relocation moves at the terminal these can be reduced. The actual amount of relocation moves that are currently being performed is known in the available data. The percentage of the amount of relocation moves being performed is 21,16%. With the help of (López-Plata, Expósito-Izquierdo, Lalla-Ruiz, & Melián-Batista, 2016) an estimation for the expected reduction in relocation moves can be made. In the paper a variety of stacking methods are tested and compared to a random stacking method. The reduction in relocation moves for the terminal will be estimated by taking the average reduction in relocation moves over the four tests performed in the before mentioned paper. With the average reduction in relocation moves being 34,24%. This resulting in the goal of bringing down the number of relocation moves at the terminal to 13,91%. This reduction in relocation moves will save the terminal time, and thus costs.

In this research the amount of ***relocation moves*** will be addressed as this is one of the more pressing problems compared to the transportation distance. The relocation problem is more adaptable to the situation at other terminals compared to transportation distances, and thus of more importance to Cofano as they want to try and implement a new stacking method application at more terminals than just the one considered in this research. The stacking problem, which often addresses the number of relocation moves at a terminal, is also widely addressed in the literature and has scientific relevance and accessible research already conducted in the field.

1.3 PROBLEM APPROACH AND AVAILABLE MATERIAL

The problem the terminal faces is that containers are currently being stacked without a computed method that can improve the stacking efficiency at the terminal. This results in more relocation moves than necessary and transportation distance problems at the terminal. In order to solve the core problem, the following stages need to be considered:

1. Devise a new stacking method for the containers.
2. Implement the solution approach that incorporates the new stacking method together with other policies to show the performance.
3. Conduct numerical experiments on the results (evaluation).
4. Conclusions and Recommendations.

Once these stages are solved, the research will be concluded, and it will be known whether a stacking method has been devised that outperforms the current policy.

The materials that are used for the bachelor assignment is what can be found in the literature concerning the subject, and the data set that has been gathered by Cofano. The data set is a large database considering various aspects of the container terminal, and all the containers moves being performed. The data covers all container information over a time span of 1 year. Data that could be used is the arrival and leave date of the containers, the import and export modalities, and the amount of relocation moves being performed. This data can be used to help create a new stacking method, and to make this new method comparable to the current policy. The knowledge from the literature can be combined with the dataset that is available to strengthen the solution that will be devised. A more elaborate overview of the available data and the data analysis itself can be found in Section 3.2.

Information can also be obtained by conversing with the company supervisor, the other employees of Cofano, and from expertise knowledge from the UT-supervisor. Furthermore, information can be obtained from previous bachelor assignments, which will provide a guideline for the research setup and to give an indication about the requirements and time limit.

The research will consist of the constructing of a solution approach that determines a method for container stacking. This method is tested whether it achieves the intended results and whether these results are applicable to the real life situation.

Together with the dataset, the literature, knowledge from the company and the restrictions of the terminal it is possible to create a stacking method that determines where to place the containers in an efficient manner. It might not be possible to incorporate all the restrictions of the terminal into the new stacking method. So, one must weigh the importance of the constraints and possibly choose to generalize in some cases. It is still the goal to make the proposed solution implementable in the real life situation, so one should be careful with what generalizations are made.

In the third stage, a numerical experiment will be conducted to ensure the validity of the model and to show whether improvements is made with considering the number of relocation moves. With this experiment Cofano can show their customers that they developed a new stacking method that, if successful, outperforms other current stacking methods. The next step for Cofano is building a software application to lease or sell to companies in which they can enter their container shipments and an application determines where to place these containers at the terminal.

Thus, the scope of the research is to develop a solution approach at the terminal that places the containers. The research is built upon reducing the number of relocation moves needed. After the solution approach has been devised, it is used to conduct experiments to show the performance compared to the current stacking policy. These results are then evaluated, and conclusions and recommendations will be given. The use for Cofano will be a solution that determines where to place containers at terminals of customers of theirs, reducing relocation moves, and thus saving costs.

1.4 RESEARCH QUESTIONS

The various parts of the report have their own research questions associated with it. The report has been divided into six different chapters that each have their own research questions and sub-questions. This division gives the following overview of the chapters in this research:

Chapter 2: What are possible solutions to the container stacking problem which can be found in the literature?

2.1 What is known in the literature about container stacking, and what concepts can be taken from it?
2.2 How can these concepts be classified to this research?

These questions can be answered by conducting a search of the literature and by assessing theses in the same field of research.

Chapter 3: How can the solution approach be shaped, incorporating the available data?
3.1/2 What can be found in the available data?
3.3.1 Which assumptions need to be considered?
3.3.2 How can one ensure the validity of the system?

These questions can be answered by conducting interviews at Cofano, by conducting numerical analysis on the data set, and with knowledge from the literature.

Chapter 4: How should the stacking algorithm work?
4.1 How will the new stacking method be shaped?
4.2 Is the implemented method of stacking valid?

In this chapter a new container stacking method is designed and implemented, build on the knowledge of the previous two chapters.

Chapter 5: What performance can be expected when implementing the different methods?
5.1 What is the performance of the different stacking methods and are the results valid?
5.2 What is the quality of the generated solution?

In Chapter 5 the results of the runs on the implemented algorithm are discussed.

Chapter 6: Conclusions and Recommendations
6.1 What conclusions can be drawn from the results of the research?
6.2 What are the recommendations to the company, and for possible further research?
6.3 What are the limitations of the research?

In the final chapter the findings from the previous chapters will be discussed. The conclusions will be assessed on the use of the research to Cofano, and to the research field overall.

1.5 CONTEXT TERMINAL

In this section the situation at the terminal will be discussed. Covering the scope and the requirements of the research which influence the solution design. The data analysis is not included in this section but given in Section 3.2.

Container transportation vehicles

In this research two container transportation vehicles are taken into account. A reach stacker that is currently being used at smaller inland terminals. And a yard crane, that is often used at larger terminals as it has fewer stacking restrictions compared to a reach stacker.

Figure 3: Reach stacker[2]

In Figure 3 a reach stacker is displayed. A reach stacked is a vehicle which drives between the yard and vessel / truck / train to load and unload containers. A reach stacker is mostly used at smaller terminals as they have stacking restrictions limiting them from reaching all containers. Reach stackers can always reach the first row of containers stacked. And can only reach the second row of containers should this row be stacked one container higher than the first row. These restrictions limit the usability of reach stackers at larger terminals, as the number of containers processed there is significantly higher.

Figure 4: Yard crane[3]

Figure 4 shows a yard crane. A yard crane is a vehicle which van move horizontally and vertically over the containers. This enables a yard crane to always reach the top container of all containers stacked in a bay (area in length and width where containers get stacked). Thus, the implementation of a yard

[2] https://www.conger.com/product/toyota-reach-stacker-container-handler/

[3] http://www.gkqzjx.com/products/Container-gantry-crane.html

crane compared to a reach stacker removes a lot of the stacking restrictions a reach stacker poses, generally increasing stacking efficiency at terminals.

Throughput

The terminal that is considered is a relatively small inland terminal. Over the timespan of 365 days a total of 18362 containers pass through the terminal. This counts down to about 50 containers a day that are processed by the terminal.

Scope

In order to devise a new method of container stacking the constraints of the terminal need to be considered. These constraints need to be implemented into the solution, as the constraints form boundaries for the solution design. After discussing with Cofano, the following constraints for the terminal are determined:

- The empty containers need to circulate to a certain degree (that the bottom one does not always remain there).
- The layout of the bay is (6 x 5 x 6) for 20 FT containers (Larger containers decrease the number of containers that can be stacked in the length of the bay). At the terminal there are 25 bays where containers can be stored.
- A reach stacker can place and retrieve containers from the first row where containers are placed. It can also place and retrieve containers over a row if the second rows' stack is one higher than in the row in front.
- Containers can be picked up from the middle (3rd row) only if they are empty until 4 high. Containers stacked in 2nd and 1st row need to be 1 and 2 containers lower respectively. If higher than 4, 1st row needs to be empty to move containers.
- The capacity of the terminal is around 2000 containers (also following from the bay size).

The scope of the terminal needs to be taken into account when designing the solution, as they restrict the ways in which the containers can get stacked at the terminal.

Requirements of the solution

To be able to create a better stacking method, a solution approach needs to be implemented. This solution approach has certain requirements that need to be fulfilled in order for it to be valid.

1. The model needs to run and output results, on which numerical experiments can be conducted in order to show the performance of the solution and to compare it to the current stacking policy. These outputs need to be checked whether they are valid and comparable to a real life situation.
2. Where possible, the solution design should be adaptable to the situation at other terminals, making it more relevant in the area of research, and more relevant for Cofano.
3. The algorithm needs to have low computational times, in order to implement it into a real time application.

Key Performance Indicators

Key Performance Indicators (KPIs) are variables that measure the performance of research or a company. In order to measure the results of this research and to make the different policies of the research comparable they will be assessed on the following KPIs:

1. The percentage of relocation moves at the terminal (the percentage of all moves that are being performed being relocation moves).
2. The occupancy rate of the terminal (percentage of the terminal filled).

1.6 DELIVERABLES

Following the core problem and other parts of the previous chapter, the following deliverables are determined:

- A method for container stacking at the inland terminal.
- An algorithm that incorporates this stacking method.
- Numerical results showing the improvement that is made to the current stacking policy and showing the validity of the system.

2. LITERATURE REVIEW

In this section, the literature that might be applicable to this research will be discussed. Papers that will be considered are papers that concern container stacking at terminals and look at how they apply to the problem at hand. As search engine Google Scholar[4], and Web of Science[5] will be used. The useful information will be taken from the papers and written down below. This information is used to answer the research question: "*What is known in the literature about container stacking, and what concepts can be taken from it?"* This question is answered in Section 2.1, after that the knowledge from the literature is related to this research in Section 2.2. This reviewed literature is mainly intended to find stacking approaches that might be applicable to this research and secondly to give a theoretical framework to the stacking problem. Also, some concepts concerning container stacking will be introduced. The final conclusions from this chapter will be summarized in Section 2.3.

2.1 RELATED WORKS

(Expósito-Izquierdo, Lalla-Ruiz, Armas, Melián-Batista, & Moreno-Vega, 2015) mention that the minimization of the number of relocations lead to an increase in storage productivity for a terminal. This is applicable to the research, as the goal is to decrease the amount of relocation moves, and as mentioned decreasing the amount of relocation moves helps improve the storage productivity.

The paper further proposes a way of stacking the containers taking into account their leave date, which is known for all containers in this case (Expósito-Izquierdo, Lalla-Ruiz, Armas, Melián-Batista, & Moreno-Vega, 2015). It might be possible to look at this method, and then adapt it to the specific situation at the terminal.

"The computational complexity of the Stacking Problem (SP) leads exact approaches to be rarely effective when tackling large cases. In this regard, heuristic algorithms have become highly competitive approaches for addressing relocation problems, providing high-quality solutions in real-world scenarios by means of short computational times." (Expósito-Izquierdo, Lalla-Ruiz, Armas, Melián-Batista, & Moreno-Vega, 2015). According to the citation above it is unwise to try and find an optimal solution for certain dimensions of the problem, rather one would try to make a heuristic that approaches an optimal solution to decrease computational times.

According to (Zanakis & Evans, 1981), heuristics are desirable and advantageous in the following instances:

- When dealing with inexact or limited data
- When a simplified model is used (then there is already an inaccurate representation of the real problem).
- When a reliable exact method is not available.
- Long computation times for the exact solution.
- To improve the performance of an optimizer.
- Repeated need to solve the same problem frequently on real-time basis.

[4] https://scholar.google.nl/

[5] https://webofknowledge.com

The authors mention more options, but these are left out as they are not relevant to this research. The paper furthermore considers the features of a good heuristic, and how to use it. So, if the option of a heuristic is chosen then (Zanakis & Evans, 1981) can provide a guideline for setting up a valid heuristic.

There are two heuristic approaches to solving a stacking problem according to (Expósito-Izquierdo, Lalla-Ruiz, Armas, Melián-Batista, & Moreno-Vega, 2015):

- Firstly, *Event-based* approaches where the block movements only occur when blocks enter or leave the system, thus rendering the stacking crane idle in the time periods in between.
- Secondly there are *Non-event-based* approaches that aims at the shortest idle time for a crane. The non-event-based approach has two advantages: the probability of further relocations is decreased, and the number of stacks with only well-located blocks is increased.

As reducing the amount of relocation moves at the terminal is the main goal of this research it is useful to look at options to implement a *non-event-based* approach.

(Dekker, Voogd, & Asperen, 2006) mention three main objectives of a stacking strategy:

- The efficient use of storage space.
- The efficient and timely transportation from quay stack and further destination (and vice versa).
- The avoidance of unproductive moves.

This with the main inputs of stacking strategies being the information available at the terminal concerning the containers. A stacking method reducing relocation moves improves a terminal on all three aspects mentioned above. So, when the research proves successful the three main objective of stacking strategies will be improved.

(Gambardella et al., 1996) mention that the main policies to be planned in a ship terminal are: how to stack containers in the terminal in an optimal manner, and how to load / unload the containers in an optimal manner. Taking this into account, the authors state that the result of the loading and unloading is strictly related to the optimal container stacking in the yard.

"A good planning system must be able to evaluate the costs of a terminal operation sequence both in term of effective costs and in term of final state of the terminal because the quality of the final terminal state could increase or decrease the costs of the following operations." (Gambardella et al., 1996). The authors mention that one should take into account both the cost of the current operation / handling that is being done, and the cost that will result from that handling. One should take into account the future cost of your current actions and look whether they outweigh each other.

(Gambardella et al., 1996) mention that the decision on where to place containers depends on many variables, with some of the more important ones being:

- The current occupation of containers in parking areas.
- The destination of containers.
- The next carrier and the best position to load the containers.
- The containers size.
- The content of the containers.

This shows that there are many variables that influence the optimal way of container stacking. It is necessary to find the most important ones and make certain generalizations for the others, as not all variables can be incorporated into the algorithm.

(Rei & Pedroso, 2011) mention that placement decisions need to be made when the following moves occur:

- Release move: When an incoming item arrives at the warehouse.
- Relocation move: when an item that is not at the top of the stack needs to be delivered and containers on top need to be removed first.
- *Remarshalling*: an unforced relocation (no container needs to enter or leave the bay), where you replace a block to prevent more future relocations.

In (Rendl & Prandtstetter, 2013) the pre-marshalling problem is addressed. What is innovative in this research is that the study considers the leave date of containers not as a set date, but rather a date range. This is an advantage over several other studies as in the real life situation the leave date of containers is also often not yet known. Since a lot of the containers at the terminal also have no set leave date it is also relevant for this research. Unfortunately, the devised method in the paper not yet outperforms the current method (Rendl & Prandtstetter, 2013).

A detailed description of generating scenarios for simulation and optimization of container terminal logistics is discussed in (Hartmann, 2004). If problems are encountered during the research it is possible to look at this paper for reference, and for ideas to continue. Further ideas can be gathered from the paper (Kefi, 2007) where a basic random stacking algorithm is set up, and then further improvement steps to the model are implemented. The random stacking algorithm is an uninformed search algorithm, and the improved method an informed stacking algorithm. The improvements are based on the leave date of the containers and the number of relocation moves being performed.

As mentioned by (Gharehgozli, Roy, & de Koster, 2016) pre-marshalling can reduce the numbers of relocations necessary at a terminal. One should pre-marshal a container when the container underneath it has a higher priority than the container on top. (Gharehgozli, Roy, & de Koster, 2016) also mention that the number of relocations at a terminal can be achieved by making a better initial stacking method. An option to reduce the number of relocations is a greedy heuristic algorithm. This algorithm places containers at an empty spot, or on a container with a lower retrieval priority. And if one of these piles is not available, then the container is stacked on a pile of containers with nearly the same retrieval time. The minimization of the relocations of a container stack while containers are getting retrieved is called a block relocation problem (BRP), which is proved to be NP-hard (Gharehgozli, Roy, & de Koster, 2016).

Loading problems deal with the storage of incoming items. Each item that arrives to the storage area must be assigned to a feasible location (Lehnfeld & Knust, 2014). Since departure times are usually unknown, the objective is to minimize the number of expected relocations (Lehnfeld & Knust, 2014). *Unloading problems* consider the retrieval of items stored until further transportation. Unloading problems considers the retrieval of items from the storage area. One decides in what order items are to be retrieved, and how to relocate the containers should a relocation move occur (Lehnfeld & Knust, 2014).

The *Loading and Unloading problem* is a combination of both. A *combined problem* both consider the loading and unloading at a storage area simultaneously. Arriving items are stored at the same time as items are retrieved from the storage area (Lehnfeld & Knust, 2014). At a terminal, containers need to be loaded and unloaded at the same time because it can occur that multiple transportation modalities arrive at the same time. Thus, a system that considers the loading and unloading problem of containers is beneficial to the solution design as it more comparable to the real life situation than just an unloading or loading problem.

Table 1 summarizes the findings from the literature:

Source	Subject	Explanation
(Expósito-Izquierdo, Lalla-Ruiz, Armas, Melián-Batista, & Moreno-Vega, 2015)	*Relocation reduction*	A reduction in relocation moves results in a higher storage productivity at a terminal improving operations.
	Event-based approach	Block movements only occur when there is an arriving or departing block.
	Non-event-based approach	Block movements occur when container arrive, depart, or when the relocation of a block can decrease future relocations.
(Rei & Pedroso, 2011)	*Remarshalling*	Remarshalling considers the relocating of containers without other containers entering or leaving the terminal. You replace the block to prevent future relocations (same principle as a non-event-based approach).
(Rendl & Prandtstetter, 2013)	-	Container get a date range rather than a set leave date. This results in a more real life comparable situation as containers often have an estimated leave date, but not a set one.
(Gharehgozli, Roy, & de Koster, 2016)	*Remarshalling*	One should relocate the container on top when the container underneath has a higher priority than the one on top.
	Greedy heuristic algorithm	Reduce relocations at terminal by placing containers at an empty spot, and when that is not possible anymore on a container with lower retrieval priority. Should this not be possible anymore, place container on one with nearest priority.
	Block Relocation Problem (BRP)	Minimizing relocations of containers while retrieving them from the bay.
(Lehnfeld & Knust, 2014)	*Loading problems*	Loading problems consider the storage of incoming items, every item that arrives must be stored reducing later relocation moves.
	Unloading problems	Problems taking into account the retrieval of containers from the terminal. It considers the order of retrieval, and the placement of containers that need to be relocated.

	Combined problems: Loading and Unloading problem (LU)	In *Combined problems* the *Loading* and *Unloading* problem are both observed simultaneously. The arriving and departing containers are loaded and unloaded at the same time. When using this method, a higher reduction of relocation moves can be achieved compared to using just a *Loading* or *Unloading* strategy

Table 1: Summarizing table literature

2.2 CLASSIFICATION OF LITERATURE TO RESEARCH

The literature gave some background to the problems that arise at terminals, and on how difficult it is to find solutions to these problems. With the help of the literature, certain pitfalls can be avoided, and it gives a structure on how to approach the problem that will be addressed in this research.

The goal of the literature search was to find literature that can be used and adapted to the situation at the terminal. From the literature search, it is found that there are many different approaches to solving the different stacking problems that arise at terminals. Most of the cases discussed in the literature consider a single aspect of the stacking problem, as it often is too complicated to solve multiple problems at once. Thus, a choice needs to be made what problem to address in this research. Since Cofano has available data about the arrivals, handling, and leaving of containers it is useful to look at a *Loading* and *Unloading* (LU) method. When considering the *loading* problem, one tries to place the containers that are loaded in a bay in such a manner that reduces the future relocation moves when retrieving them. When the containers are correctly being placed this will result in a decrease in relocation moves, and thus optimizing processes at the terminal. When considering the *unloading* problem, one retrieves the containers from storage, and when the relocation of containers is necessary, the containers get relocated minimizing the expected future relocations. Now, when containers simultaneously get loaded and unloaded, the *Loading* and *Unloading* (LU) problem is addressed.

What is known in the literature about such loading and unloading problems is limited. A wide variety of papers take into account the loading or unloading of vessels. But few consider the combination of both problems. The most widely applied method is assigning a priority (usually leave date or expected leave date) to a container, and stack / relocate them according to this leave date. This method of stacking ensures that containers that are estimated to leave at an earlier date are placed on top of containers that are estimated to leave later. With the available data it is possible to assign an expected leave date to the containers, making this approach feasible for this research. Since LU problems have not been widely addressed in the literature this research contributes to the database of literature concerning container handling operations.

The computational complexity of stacking problems, or for that manner loading and unloading problems, can get quite high. In case the computation times for the problem get too high, a choice needs to be made to use a heuristic instead of an exact solution approach. A heuristic provides a feasible solution but does not certify the optimality of it. As mentioned in the section above, there are several instances when the use of a heuristic is preferable. Thus, when it appears to be the case that a heuristic is a better approach for this research, the implementation of a heuristic needs to be taken into consideration. When applying a heuristic, computational time decreases, while still finding a

feasible solution. And as the final goal of Cofano is to have an application running in real-time computational times might get too high to handle when applying an optimal solution approach.

In this research it will be better to use a *non-event-based approach*. When using a non-event-based approach, the number of relocations is further reduced because of the container replacements in between outgoing and incoming blocks, when transportation modalities would be idle. As it is possible to give an estimated leave date one can remarshal the containers according to this date. However, since the research already takes into account the setting up and assessing the performance of an entire new stacking method, an *event-based* approach will be used, as the implementation of a *non-event based* approach is not feasible within the scope and time of this research.

There are many variables that influence the design of the solution approach, and the solution approach itself. One needs to consider these variables and see what generalizations need to be made for the process that is chosen to be optimized. Thus, the generalizations that need to be made for the loading and unloading process must be chosen such that they do not majorly influence the outcomes of the research. As the validity of it needs to be ensured.

As the terminal processes are hard to optimize in a generalised manner, the research might need to scope down more to the situation at the terminal. Making the problem easier to solve, but harder to adapt to other instances, and less relevant in the field of research.

2.3 CONCLUSION

Concluding the literature review, the goal was to learn what was known in the literature about the container stacking problem, and what could be taken from it and applied to this research. The literature review introduced a lot of concepts that gave background knowledge to the stacking problem. Furthermore, the literature review contained an analysis of these concepts and how they are applicable to this research.

The following things were concluded in the literature review:

- The problem considered in this research belongs to the class of Loading and Unloading (LU) problems.
- An *event-based* approach will be used.
- Examples of models / heuristics previously used, that can give insights into pitfalls and possible solutions.
- A heuristic approach might be needed, to ensure that the model has a feasible solution.

With this knowledge from the literature the solution design can be set up. Various aspects from the literature can be combined to help formulate a model which corresponds to the real life situation at the terminal.

3. SOLUTION APPROACH

In this chapter the following research question will be answered: *"How can the solution approach be shaped, incorporating the available data?"*. In order to answer this question firstly the method of data gathering is explained. In the Section 3.2, the gathered data is analysed and conclusions relevant for the solution design taken. In Section 3.3, the results of the data analysis are incorporated with previous sections of this report to shape the solution design.

3.1 DATA GATHERING

In this section the data analysis will be discussed. The data provided by Cofano was a database which needed to be loaded into pgAdmin4[6] in order to be able to find the applicable data. PgAdmin4 is an open source administration and development platform. Firstly, the relevant parts of the database needed to be connected to each other. The relevant parts of the database being the bookings of the containers, the associated actions with these bookings and the handlings performed on these actions. These databases were linked using PostgreSQL[7], which is a database management system incorporated into PgAdmin4. Once these three parts of the database were connected, relevant information was selected and then exported to Excel. In Figure 5, the database schema of the relevant parts of the database and their connection are displayed.

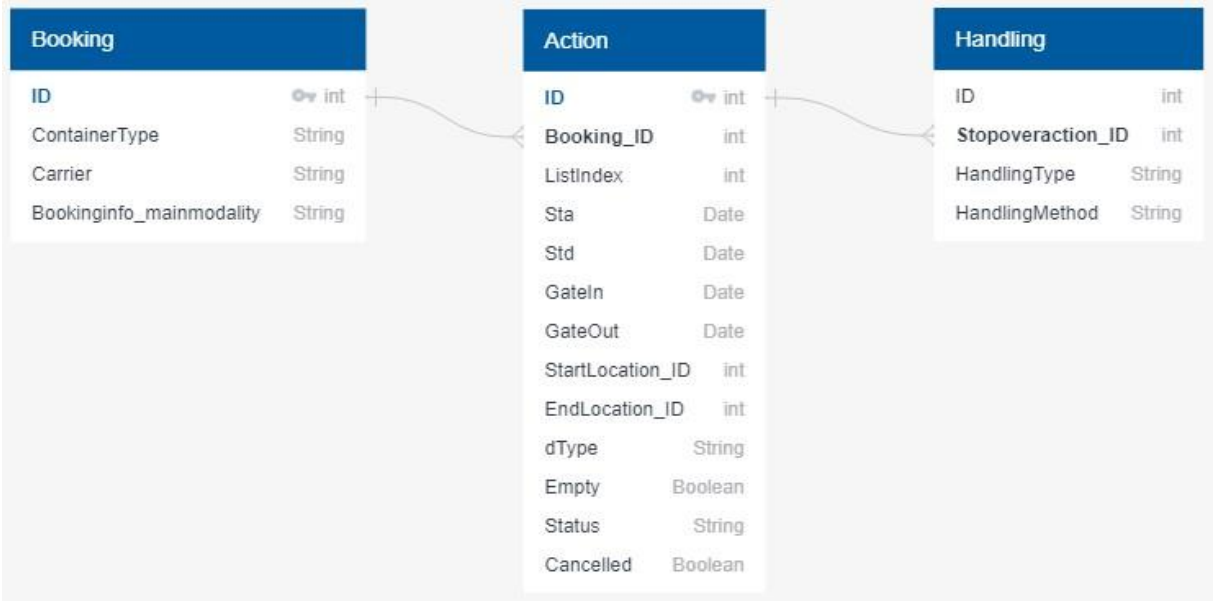

Figure 5: Database Schema used to obtain data in pgAdmin 4.

By using different queries, the relevant parts of the database are gathered and exported to Excel where the data is analysed.

Data validity

There are a few issues with the data that is obtained from the database. There is some incorrectness and incompletion. The data has been gathered by employees logging their own actions, which is the cause for the incorrectness in the data because of human error. This incomplete and incorrect data

[6] https://www.pgadmin.org/

[7] https://www.postgresql.org/

needs to be filtered from the other data. Mainly, the data considering the time it takes for a move to be completed is incorrect, as the terminal workers themselves need to log this and do not consistently do it properly. The most common time it took for a relocation move (moving a container on top of a container that needs to be retrieved and relocate it) was 1 second, which is not possible. Because of this, the data taking into account the time it takes for a move to be performed is considered incorrect and will not be used in this research.

The incomplete data differs per different data type (e.g. not for all containers an estimated arrival and leave date is known). So rather than making one dataset of complete data, the dataset will be altered on different criteria (e.g. when doing tests on the arrival and leave data of containers include all data that is complete on this criteria). This results in various smaller sets of data, rather than one large set.

So, the data is pre-processed, with the aim to remove the following inaccuracies from the different data sets that will be used: Lack of available information, unfeasible transportation times, and small numbers of special cases irrelevant to the complete dataset.

3.2 DATA ANALYSIS

In this part the gathered data from Section 3.1 will be analysed. The different datasets used in this research incorporate: the movement of the containers throughout the terminal, the different container types and sizes, the estimated arrival and leave date of containers, and the relocation moves that are being performed at the terminal.

Traffic data

In the traffic data the general movement of containers at the terminal is analysed. This includes the number of different containers, moves, and the different container types that pass the terminal. The number of container moves at the terminal are displayed in Table 2.

Move	Nr. of moves	% total
Relocations	9253	21,16
Barge	17437	39,87
Truck	17045	38,97
Total	43735	100,00

Table 2: Container movement at terminal over 364 days.

Table 2 shows that 21,16% of the total number of moves at the terminal are relocation moves. As mentioned in Section 1.2, the goal is to reduce the number of relocation moves at the terminal to 13,91%. It can also be seen that the number of barge moves, and truck moves are quite similar. This is logical as most containers arriving by barge are transported by truck to be unloaded somewhere and then transported back by truck and barge again. Furthermore, Figure 6 shows the different containers types that are handled at the terminal and the quantity at which they occur. In Figure 6 only 7 out of 17 container types are displayed, as the 10 container types with the least containers processed by the terminal only make up 1.34% of the total number of containers. Thus, these last 10 container types are not taken into account as their numbers are insignificant to incorporate into the research design.

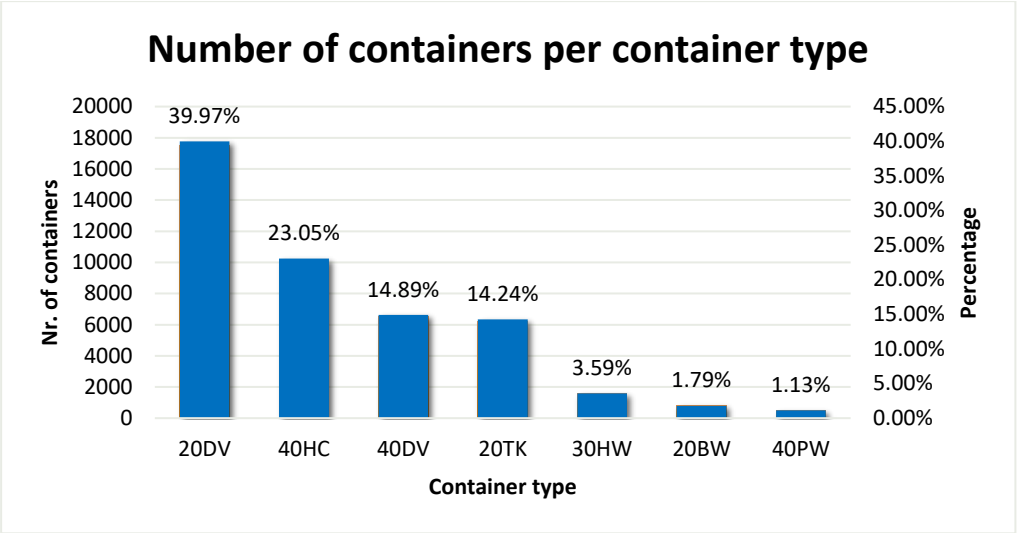

Figure 6: Number of containers per container type over 364 days.

In Figure 6, it can be seen that mostly 20 and 40 Ft containers are processed at the terminal with a relatively small number of 30 Ft containers. The different container types are relevant as they have repercussions for the number of containers that can get stacked in a bay. In Table 3 the individual properties of these container types are displayed.

Type	Size (Ft)	Specificity
20DV	20	Standard type
40HC	40	30 cm higher than DV
40DV	40	Standard type
20TK	20	Tank Container
30HW	30	30 cm higher and 10 cm broader than DV
20BW	20	N/A
40PW	40	10 cm broader than DV

Table 3: The different container types with their sizes and specificities.

In Table 3 the different containers types with their specificities can be found. When looking at Table 3 and Figure 6 combined, it can be seen that most containers that are processed by the terminal are containers of the standard type. The different specificities of the container types bring no additional stacking constraints, as they can all be stacked onto each other. The only constraint that comes from the different type of container is the size. A 40 Ft container can get stacked on top of two 20 Ft containers, but not the other way around.

Estimated arrival and leave date of containers

The most important aspect of the data that is available is the estimated arrival and leave date of containers, and the comparison between these dates and the actual arrival and leave dates. With this data, priorities can be assigned to the containers, and the containers can then be stacked according to this priority. As not all the containers have an estimated arrival and leave date, just the containers that have complete information on these dates will be considered. This results in a dataset of 3023 containers over a time horizon of 364 days. Of these containers it is also known whether they were transported with a truck in between the arrival by barge and leave by barge, and by which carrier they were transported.

Estimated	Containers	Percentage
Earlier	2670	88%
Later	353	12%
Total	3023	100%

Table 4: Containers that left before or after their estimated leave time.

As can be seen in Table 4, most containers (88%) leave before their estimated leave date. Only 12% of the containers leave after their estimated leave date. This means that when containers get stacked according to their estimated leave date, they probably leave the system earlier than expected. In Table 5, a comparison between the average estimated time until leave and the actual time until leave is displayed. The time barge in until barge out gives the hours that a container is expected to, and the actual time between, arrival by barge and leave by barge. The same estimate is displayed for the hours until truck move, only the actual time until a truck move is not known, as the data does not incorporate this aspect.

Average	Time barge in until barge out (h)	Time until truck move (h)
Terminal estimated	101:44:21	66:43:13
Terminal actual	94:54:50	N/A

Table 5: Comparison average time at terminal

From Table 5, it can be seen that the average time a container spends at the terminal (from the arrival by barge and the leave by barge) is quite accurate. On average the containers leave earlier than planned, as displayed in Table 4, and this also shows in Table 5. On average containers leave the terminal by barge about 7 hours earlier than estimated. This can be explained by the fact that the time of terminal departure of the container is logged once the container enters the barge. However, the barge does not immediately leave as more containers need to be loaded, causing some difference between estimated times and actual times. Furthermore, it can be seen that the estimated time until a truck move (truck picking up container, unloading it at company, and then returning it to the terminal) is lower than the container time at the terminal. Which is obvious, as the *'time barge in until out'* must be larger because most containers are picked up and transported by truck in between their barge moves. As to why a large part (88%) of the containers leave before their estimated leave time is that containers when estimating their barge leave time are always estimated at 15:00. As a normal working day takes place more hours before than after 15:00, most containers leave before this time. The containers are estimated in this manner as the time of arrival of a barge is uncertain, so they are rather estimated on a leave date than a leave time.

Below in Figure 7 and 8 two histograms are displayed. In Figure 7 a histogram of the containers which leave earlier is displayed. It shows the interval of how much earlier the container leaves the terminal than estimated. The same applies to Figure 8, just now for the containers that leave later than their estimated time at the terminal.

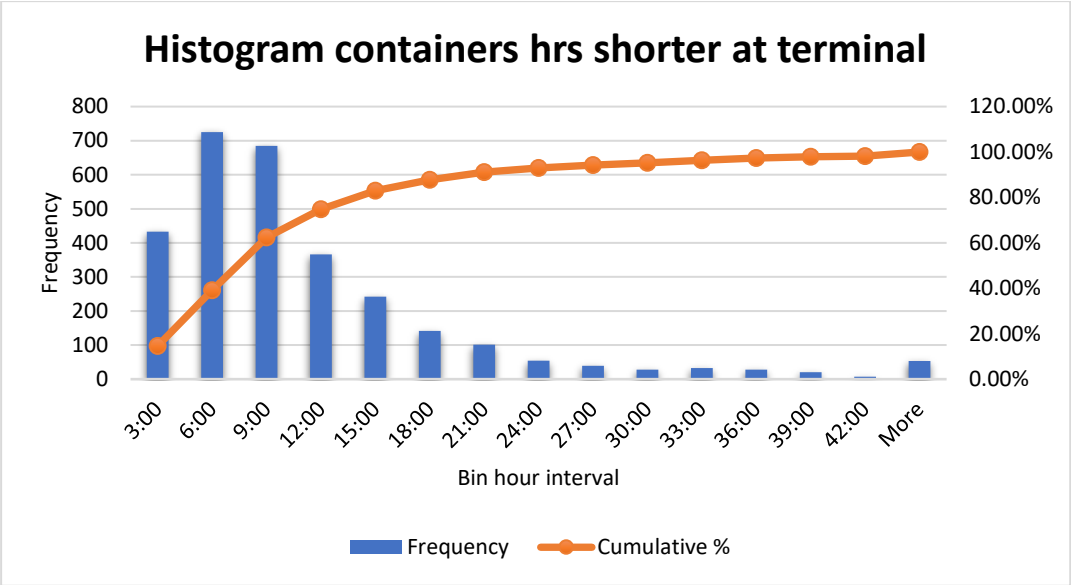

Figure 7: Histogram of containers staying shorter at the terminal than estimated.

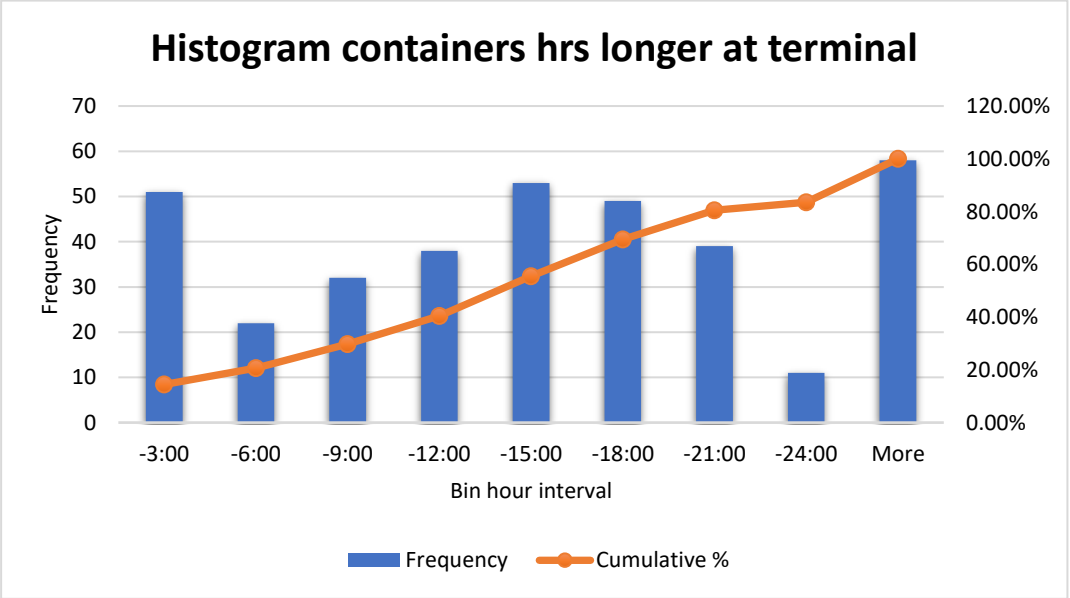

Figure 8: Histogram of containers staying longer at the terminal than estimated.

Figure 7 displays the containers that leave the terminal earlier than estimated. Most containers leave [0:00 – 12:00] hours earlier. This indicates that the containers that leave earlier largely still leave on the same day as they were planned. The same holds for the containers that stay longer at the terminal than estimated, as can be seen in Figure 8. The difference in time that the containers stay longer or shorter at the terminal can thus also be explained by the fact that the arrival and departure times of a barge are estimates, and not precisely timed on the hour. Thus, it is safe to assume that the estimated times of arrival and departure are largely accurate to the day of arrival or leave.

Status	Barge	Barge	Truck	Truck
Estimate date	173	50%	6	2%
No estimate	174	50%	243	98%
Total	347		249	

Table 6: Number of known estimated leave date, and unknown.

In Table 6 it can be seen that for barge movements for 50% of the containers it is known beforehand when they need to leave again (an estimated leave date which largely corresponds to the actual leave date). However, for the truck movements the estimated leave date is 98% of the time unknown as can be seen in Table 6. The containers that have no information about their estimated leave date need to receive their own estimation based on the historical data. For the retrieval of the percentages in Table 6 only containers which have a “Next” or “Not_Started” status have been considered. This because containers that have already been processed by the terminal and receive a “FINISHED” status receive an Etd (Estimated departure date) of the date that they have left the system and is thus wrongly displayed in the data set. Only containers that are yet to arrive to the terminal with an estimated leave date are relevant for the stacking of the container.

This estimation of containers with an unknown estimated leave date is done by taking the average hours a container stays at the terminal per carrier. Figure 9 displays these averages. The names of the carriers have been replaced with numbers to ensure the data confidentiality. The carriers that have transported 1-32 containers at the terminal over a time span of 364 days are disregarded in the data, as these numbers are too small to make their average viable. The containers that are transported by an unknown carrier will be given an estimated leave date according to the average time of the other carriers (1-13).

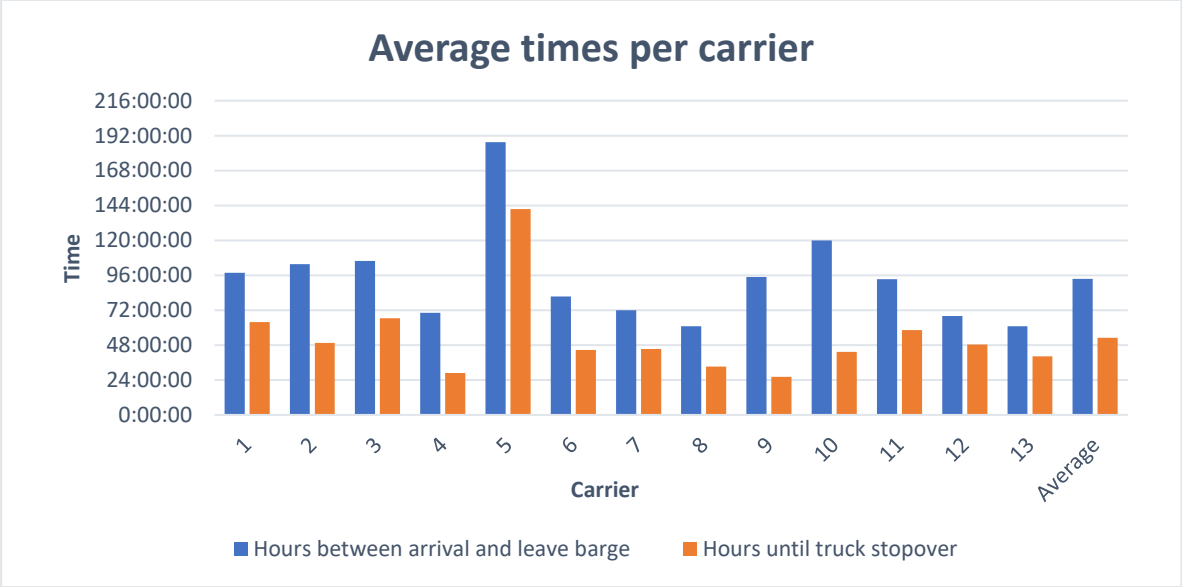

Figure 9: Graph showing average time a container stays at terminal per carrier.

For example, when a container of carrier 3 arrives to the terminal, the estimated leave date of this container will be about 72 or 96 hours, depending on whether there is a truck move or not. Figure 9 shows that there is a difference between the estimated arrival and departure time of a container per different carrier. Thus, it is beneficial for the algorithm to consider the containers of their designated carriers with their corresponding times at the terminal. To clarify, the time arrival and leave barge is the time between the estimated arrival of a container at the terminal and the estimated leave by barge. The time until truck stopover is the time it takes on average between the arrival by barge and a possible planned truck move. Most containers arrive to the terminal, stay there until truck transportation, and then leave by barge again. Thus, most containers are transported by truck in between the arrival and leave by barge. Explaining why on average the hours until a truck stopover is shorter than the hours

between arrival and leave barge. With these averages, an estimated leave date can be given to the containers that as of yet have an unknown one, and these containers can then be stacked according to this estimated leave date.

Number of relocations

Another aspect that can be found in the data is the number of relocations that are performed. For each day it is known how many containers arrive, leave, and are relocated at the terminal. This information shows us the number of relocation moves compared to the number of containers entering and leaving the terminal per day. The data is shown in Figure 10.

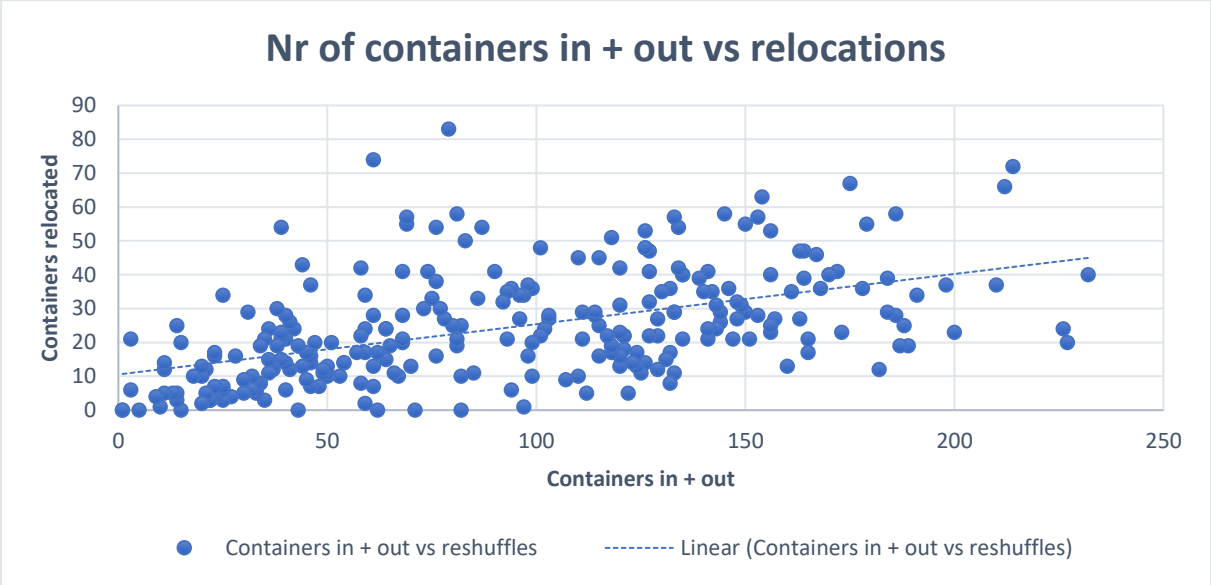

Figure 10: Graph showing the containers entering and leaving the system on that day vs the number of relocations on that day.

In Figure 10 it can be seen that when the number of containers entering and leaving the terminal increases, the overall number of reshuffles performed also increases. Although the increase is not 1:1, as can be seen from the trendline, the relative number of relocation moves decrease when the total number of containers in and out increase.

Average	in	out	reshuffles
Containers / day	42.42	49.72	24.23

Table 7: Average container movements per day.

In Table 7 one can see the number of containers that are processed at the terminal each day on average. It can be seen that on average there are more containers moving out of the terminal than into the terminal. This difference can be explained by the inconsistencies in the data set, and by containers that have not been fully processed at the terminal yet. But it can be assumed that all container that enter the system must also leave it again, thus the inflow of containers must be equal to the outflow of containers.

3.3 SOLUTION DESIGN

In this section the solution design will be explained. The previous parts of the report come together to create the solution design. With the solution design being what needs to be incorporated into the solution such that it reflects the real-world situation, and what can be incorporated into the solution

that will improve the quality of it. The approach to setting up an algorithm is mapped in order to derive the algorithm in Section 4.

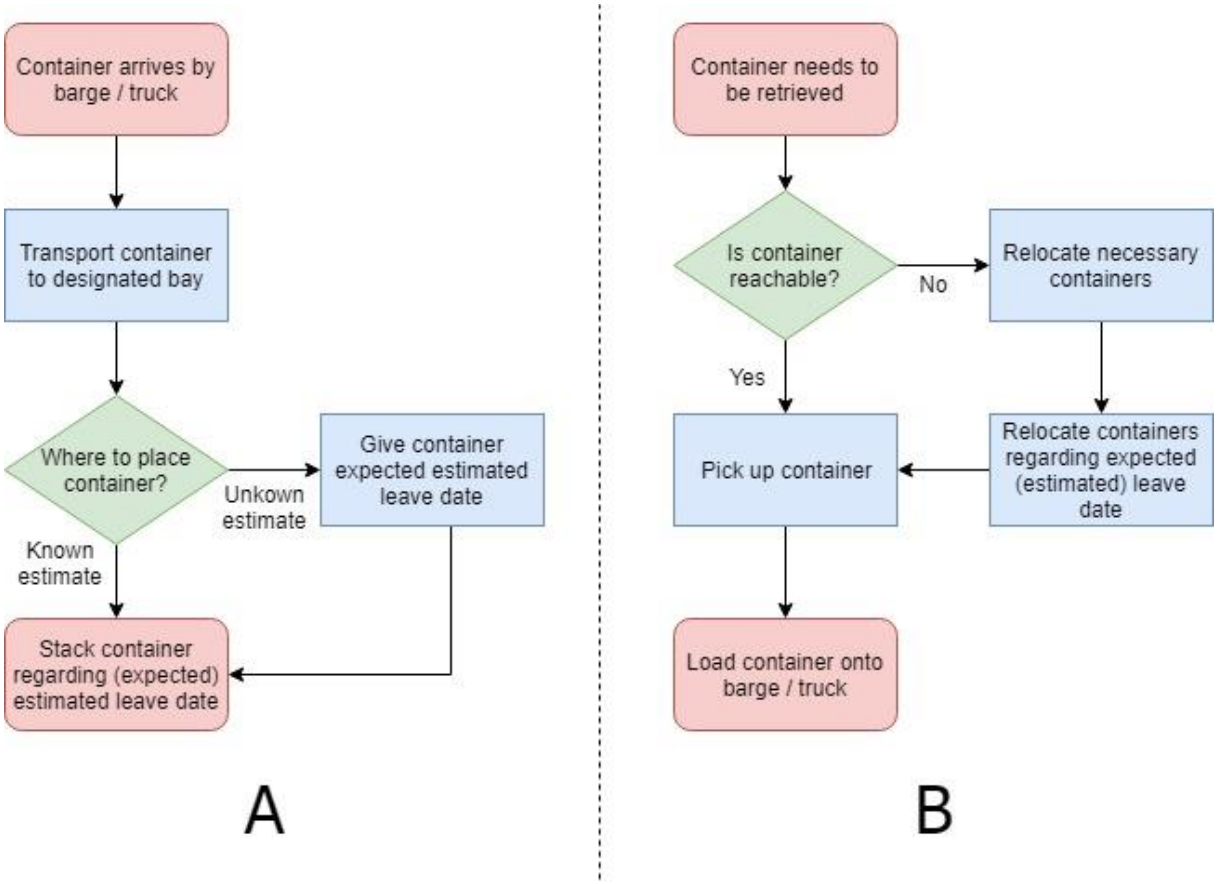

Figure 11: Flowchart displaying the loading and unloading of containers at the terminal.

Figure 11 shows the process of a container arriving to or departing from the terminal. In process A, a container arrives to the terminal by truck or barge and needs to be transported to the bay where it needs to be placed. After the container arrives at the bay, the place of stacking needs to be determined. When the container has a known estimate leave date (Etd), the containers can get stacked according to this leave date. The containers that do not have an estimated leave date need to get one assigned. This can be done by assigning these containers estimated leave dates based on the historical data of the corresponding carrier transporting the container. The average times a container spends at the terminal corresponding to their carrier is shown in Figure 9.

Furthermore, the process of a leaving container is also mapped in Figure 11, process B. When a container needs to leave the system it first is checked whether other containers are stacked onto the container which needs to be retrieved. If it is the case that containers are stacked on top of the one that needs to be retrieved, these first need to be relocated.

Another case where containers need to be relocated is when the reach stacker cannot reach the containers because of blocking containers in the row before or after. A reach stacker can retrieve containers over a row if the back row is stacked one higher. So, containers in the row before the row from which a container needs to be retrieved must be one lower respectively. Otherwise these containers also need to be relocated. The same applies to containers stacked two rows before or after the containers in the retrieving row. If this appears to be the case, then all containers from this row

need to be relocated. The relocating of containers also occurs in accordance to the estimated leave date of the container.

After the new stacking method is devised, and the possible generalisations are made, the writing of the algorithm can start. As most of the literature reports, it is quite hard to make a model that provides the optimal solution, as it has long computational times and is often complex. So, if this appears to be the case, a heuristic will be made to come up with a good solution. The devised algorithm considers the placement and relocation of the containers at the terminal reducing the total number of relocation moves.

All the constraints and specificities of the terminal need to be translated into the algorithm. Possibly algorithms or heuristics from the literature can be used and adapted to fit the situation at hand. The algorithm should be able to run and produce numerical results. After the algorithm is made and the results from the experiments are analysed, the performance and validity of the devised method will be known.

Validation

In order to ensure that the results from the model can be trusted, some form of validation is needed. If the model is not validated one cannot assume its outcomes to be true. In this research the model will be validated by means of 'black box validation'.

Black-box Validation

In black-box validation one looks at the results / data from the algorithm and compares it to the expected results stated before the experiment. It does not test the code itself of the algorithm, but just the outputs from the experiments with the finished algorithm. The expected outputs are set up according to the selected inputs into the experiment. The requirements for and the specificities of the algorithm are the foundation for the test cases. The goal of black-box testing is to determine whether the model is working as expected, and whether it meets the expectations set for it (Lozancic, 2017). Black-box validation is important for this research. It is used to ensure that the final outputs of the model are not illogical, ensuring validity.

3.4 CONCLUSION

In this chapter we wanted to find a design for the solution according to the data that is available and the situation at the terminal. The research questions that have been set up in Chapter 1 have been answered.

The first research question *"What can be found in the available data?"* has been answered. From the data that has been gathered by Cofano useful conclusions can be drawn. Aspects that will be useful to implement into the algorithm are the estimated departure dates of the containers, and the analysis of the historical data to give an estimated departure date to the containers that are as of yet unknown. Corresponding with this estimated departure date, the containers can get stacked at the terminal with the aim to reduce the number of relocation moves that are necessary. Furthermore, the data makes it possible to compare the current stacking policy to the method that will be devised in this research. The previous is important as Cofano wants to show their customers that the new method will bring improvement compared to their current policy. This in order to be able to lease a possible application to said customers

Furthermore, in the Solution Design in Section 3.3 an approach for determining a new method of container stacking is given. This method will be tested with the implementation of an algorithm. The method stacks the containers according to their (given) Etd, ensuring that containers that are estimated to leave at an earlier date are stacked on top of containers that are estimated to leave at a later date. Lastly, some options for validating the algorithm are given. It is important to validate the algorithm that is devised, because otherwise one cannot assume its outputs to be credible.

4. NEW STACKING METHOD

In Chapter 4, the new stacking method will be devised and implemented. The research questions that will be answered in this chapter are: *"How will the new stacking method be shaped?"* and *"Is the implemented method of stacking valid?".* The answers to these questions can be found in Section 4.1 and 4.2, respectively. In Section 4.1, the different stacking policies are introduced and following in Section 4.2, the implementation of them is described. In Section 4.3, the conclusion and answers to the research questions are given.

4.1 ALGORITHM

This section described the algorithm that has been developed for the placement of containers at the terminal. The steps that the algorithm takes are described, together with the assumptions that are needed in order to be able to construct the algorithm.

Approach

The goal of this research and thus of the algorithm is to optimize container stacking operations at the terminal. The optimization of container stacking operations will be done by means of constructing an algorithm that determines where to place the containers at the terminal. An algorithm is *"a set of mathematical instructions or rules that, especially if given to a computer, will help to calculate an answer to a problem"* [8], thus by programming an algorithm a solution can be computed for the container stacking problem.

The problem that is addressed in this research is the absence of a methodological stacking method. The algorithm will focus on minimizing the number of relocation moves at the terminal. Recall from the list of definitions that a relocation move is a move that relocates a container that is blocking the retrieval of another container. Relocation moves at a terminal are unproductive moves and all the time spend on them are wasted resources and time.

When constructing the algorithm, the containers are ordered and stacked based on their Estimated departure date (Etd). Their Etd is known, as the historical data of already processed containers is used as input. In the real life situation not all the Etd's of containers are known. When this is the case, the average time a container spends at the terminal of their corresponding carrier can be used. As the algorithm does not consider the uncertainty of the leave date the results when applied in a real life situation might differentiate from our results. But as most Etd's of containers are accurate it will not have a large influence on the results. Another prerequisite for the working of the algorithm is that all containers receive an Etd.

For the algorithm the following 3 processes are defined:

1. The placement of arriving containers.
2. The retrieval of containers.
3. Special case: The processing of containers with a truck move.

In order to measure the performance of the stacking algorithm multiple policies will be set up. In this manner the performance of the devised algorithm can be compared, and it shows whether

[8] https://dictionary.cambridge.org/dictionary/english/algorithm

improvements are made compared to the current situation. The following policies are used to assess the performance of the different stacking methods:

1. A random policy
2. The current stacking policy
3. The new stacking method
4. Implementation of a yard crane using the current stacking method
5. Implementation of a yard crane using the new stacking method

A random policy will be used in order to provide a baseline for the other policies and to assess whether the stacking policies are valid. When the stacking policies do not outperform the random stacking policy one cannot assume the model to be valid, as a random stacking policy should have the worst performance. Furthermore, the current stacking policy, the new stacking policy, and the possible implementation of a yard crane are assessed. Together with the baseline of the random stacking policy it is possible to compare the performance of these three stacking policies. Currently the terminal uses reach stackers for the transportation of their containers, but a yard crane outperforms these. For a small inland terminal, the acquisition of a yard crane might not outweigh the cost. However, the yard crane option might be viable when implementing the stacking method devised in this research at larger terminals. The performance of a possible yard crane implementation is assessed with the current and new stacking method. This in order to compare the performance of the new and current stacking method when using different container stacking vehicles. Pictures and explanation on the workings of reach stackers and yard cranes were included in Chapter 1 Section 1.5.

All stacking and relocating occurs under the constraints of their designated transportation modality. With the stacking constraint of the reach stacker being that it can stack and retrieve all top containers in the first row where containers are stacked, and if the container location is one higher than the front row the reach stacker can place and retrieve containers in the second row where containers are placed. So, the maximum rows the reach stacker can reach is 2, under the condition that the second row is stacked one higher than the first. Furthermore, there is the implementation of a yard crane. A yard crane can reach every top container located in a bay, as it can move horizontally and vertically over the bay.

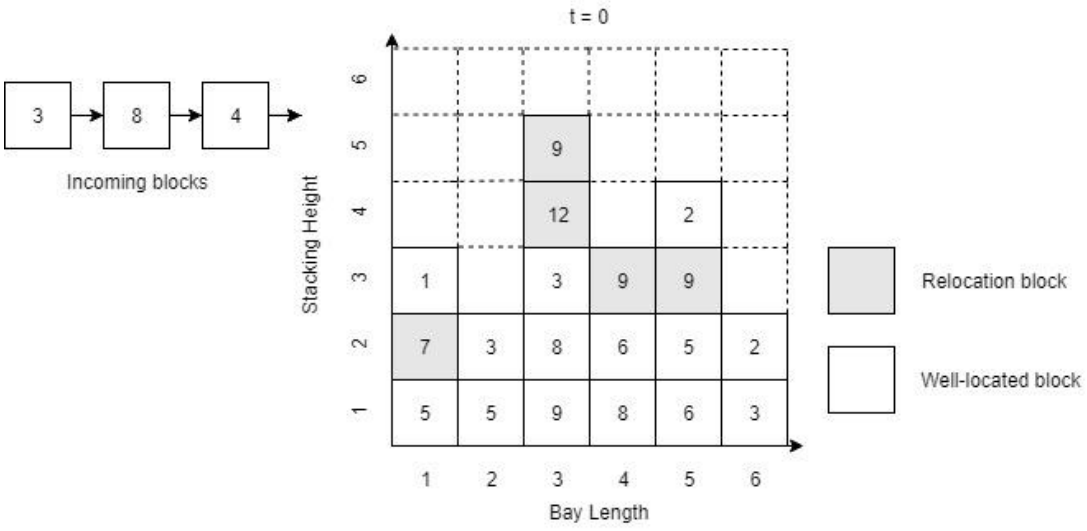

Figure 12: 2D Block Scheme container movement

Figure 12 displays the stacking situation for a single row in a bay. The X-axis displays the length of the bay, and the Y-axis the stacking height. The numbers represented in the container blocks are their corresponding Etd's. Each day their Etd's decrease by 1, and once their Etd reaches 0 the blocks are retrieved. The grey blocks represent blocks that are not well placed and need to be relocated in order to retrieve the underlying blocks that need to leave the terminal at an earlier date. In the case of the usage of a reach stacker, a well-located block might also need to be relocated in order to retrieve a container from the row before or after. The block might be well-located within the row, but still block the retrieval of a container from another row under the stacking constraints of the reach stacker.

Random stacking policy

As mentioned before, a random stacking policy will be made to provide a baseline for the performance of the other stacking methods or policies. In the random stacking policy, arriving containers are located randomly in the bay. Furthermore, when a relocation move is needed because of a blocking container this relocating also occurs randomly. The placement and relocating of containers are executed under the stacking constraints of the reach stacker.

Current stacking policy

In the current stacking policy containers are put together in clumps corresponding to the carrier who owns the container. The current stacking policy is implemented in order to be able to compare it to the other stacking policies. The current stacking policy algorithm tries to place the containers that arrive to the terminal on top of containers of the same carrier. When this is not possible the containers are placed on an empty ground level. Should all ground levels be filled, then the containers are stacked on top of a container with a different carrier. The same method is applied when relocating the blocking containers. The placement and relocating of containers are executed under the stacking constraints of the reach stacker.

New stacking method

In this section the design of the new stacking method will be explained. Subdivided in the three different processes mentioned before: the placement of containers, the containers with a truck move, and the retrieval of containers. The new stacking method is based on the greedy stacking heuristic proposed in the literature review by (Gharehgozli, Roy, & de Koster, 2016), with some refinements possibly improving stacking efficiency. In Figure 13 the workings of the algorithm are mapped in a flowchart. With the explanation of the workings of each process following.

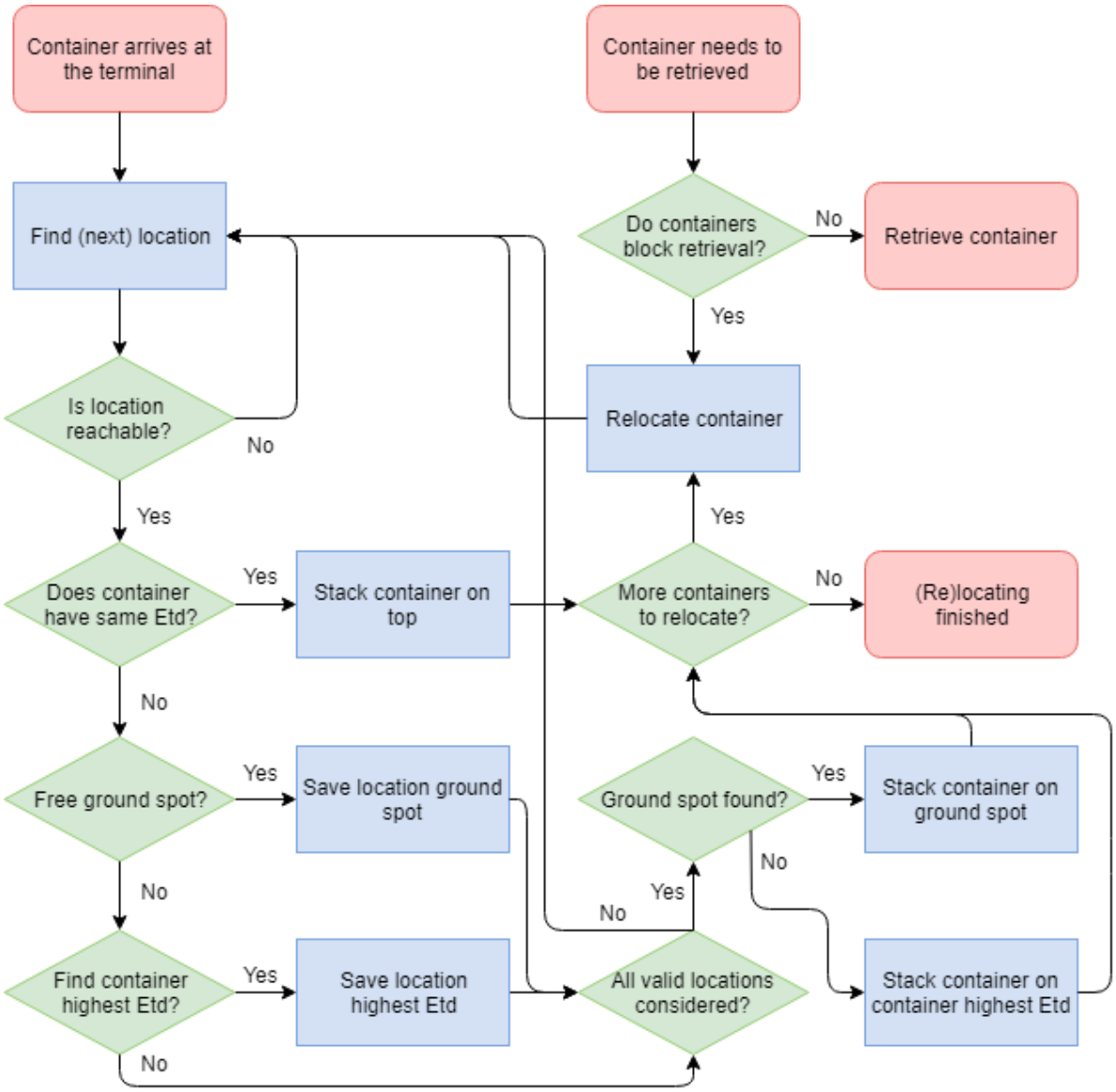

Figure 13: Flowchart displaying workings of algorithm

Process 1: The placement of arriving containers

The first step of the algorithm is to stack the arriving containers based on their estimated departure date (Etd). When the containers arrive to the terminal and already have an Etd they can be stacked according to this Etd, as can be seen in Figure 13. The containers that do not have an Etd will receive an estimated Etd based on the average time the containers of their corresponding carrier stay at the terminal. These average times can be seen in Figure 9. These times will have to be rounded to days for the algorithm to be able to handle them. When stacking containers with the same Etd on top of each other one ensures that all these containers need to leave simultaneously, improving stacking efficiency when stacks of the same Etd containers arise.

When all containers have their Etd, the stacking can begin. The containers are stacked on top of each other when they have the same Etd. And should no such stack occur, then they are placed on the ground level. This improves the stacking method, as stacks arise with containers that need to leave on the same day ensuring more free ground spots for containers with a different Etd. The first container is placed in the most right spot of the 5th row, and when filling ground levels this occurs from the right

to left and then from the 5th to 1st row. It is irrelevant from which side the stacking occurs, as in the most ideal solution all container fit in three rows to decrease the number of relocation moves caused by the stacking constraints of the reach stacker.

Once the ground levels of the bay have been filled up the arriving containers will be stacked on top of the containers with the highest Etd (the container that needs to leave the terminal the latest). This ensures that the containers can stay at their designated spot for a longer period and increases the probability that the containers stacked on top of the container with the highest Etd will leave earlier, thus reducing the number of relocation moves. The placement of containers is executed under the stacking constraints posed by the reach stacker.

Process 2: The retrieval of containers

The second step of the algorithm considers the retrieval of containers stored at the terminal. When the containers reach their Etd they are picked up for transportation from the terminal. This transportation occurs by barge or by truck. Containers that need to be picked up and are stored on the top of a stack and have no blocking containers in the rows before and after can be picked up immediately without any relocation moves. However, for containers that need to be retrieved from the terminal and still have containers blocking this retrieval more moves are needed.

The containers that are stacked on top of a container that needs to be retrieved need to be relocated within the bay. This relocation can occur on the Etd all the containers at the terminal have or have received at arrival. The relocating of containers occurs in the same way the stacking does. Firstly, containers are stacked on containers with the same Etd. When this is not possible, they are relocated to an empty spot, and thirdly they are placed on top of the container with the highest Etd. All restacking also occurs under the stacking restrictions posed by the reach stacker.

Special case: The processing of containers with a truck move

Containers that need to be transported by truck need to be processed by the terminal twice. They enter the terminal by barge, are retrieved and returned by truck, and then transported back by barge again. When such a container enters the terminal, the Etd of this container is the estimated time until truck transportation. It is assumed that a truck retrieval and return always happens on the same day.

Furthermore, when containers leave by truck transport, they are always loaded or unloaded in between. So, container status about the load of the container changes after truck transportation. Empty containers become full, and full containers empty.

Conclusion

When these three processes are incorporated into the algorithm, the new stacking method is devised, and it makes it possible to conduct numerical experiments on the algorithm showing the performance of it compared to the performance of the random and current stacking policy. The algorithm is based on a greedy stacking heuristic mentioned in the literature by (Gharehgozli, Roy, & de Koster, 2016), with some added refinements. The proposed heuristic in the literature starts with placing containers at ground level, quickly filling up all ground levels. As it is more beneficial to stack containers with the same Etd on top of each other, the newly devised stacking method firstly considers this, before stacking on ground level. Certain aspects could still be included into the algorithm, but these will not be

addressed in this research. These aspects are mentioned in the recommendations for future research in Section 6.2.

Yard crane implementation

In order to expand the scope of this research and to widen the possibilities for implementation at other terminals, the option of implementing a yard crane is examined. The performance of the yard crane will be assessed by using the new and the current stacking method mentioned in the previous sections, only without the constraints of the reach stacker. A yard crane can reach all top containers located in a bay without the row restrictions a reach stacker has. This makes yard cranes overall better at terminal operations, reducing relocation moves as it can always reach all the stacks located in the bay.

4.2 IMPLEMENTATION

To analyse the performance of the proposed algorithm the algorithm is implemented in Excel using Visual Basics for Applications (VBA). The version of Excel where the algorithm is implemented is Microsoft© Excel© for Office 365 MSO (16.0.11929.20234) 32-bits (Version 1908). This implementation in Excel enables us to analyse the probable performance of the algorithm should it be implemented in a real life situation. In this section it is explained per step how the implementation is done. Furthermore, there is a list of assumptions included which were needed to develop the algorithm.

Input data

As input for the algorithm, three different sized datasets will be used. These three sets contain the historical data of container movement at the terminal. The data that is used consists of the day of arrival, day of departure, and day of truck transportation of the containers. The size of the complete dataset consists of 3054 containers over a 156 day period. From this large dataset, three different sized datasets are composed consisting of 3000, 2000, and 1000 containers respectively. The selecting of the containers for each dataset was done randomly from the total number of containers in the database of 3054 containers. Note that the total number of containers arriving to the terminal is not the number of containers in the database, as containers with a truck transportation move through the algorithm twice. This makes the number of arriving containers to the terminal 5227, 3461, and 1709 respectively. All these containers are run over the one bay considered in the algorithm.

In the pseudocode sometimes counters are used. These counters are necessary as they loop over the virtual bay visualised in excel and save the relevant container information. The irregularity in these numbers is because the counters represent certain cells in Excel that represent the visualisation of the bay layout shown in Appendix B.

Assumptions

In order to be able to develop an algorithm the following assumptions are made:

- Containers can only be (re)stacked in one bay.
- The estimated arrival or departure date of the container is also the actual arrival or departure date in the algorithm. Which can be assumed because this is the case for the larger part of the containers.
- All containers have a known Etd.

- Containers that need to be placed in areas designated for dangerous goods are not processed by the algorithm.
- Truck retrieval and return happens on one day.
- Containers of different sizes get stacked in different bays.
- All the containers that arrive on a day are placed before retrieval starts.

New stacking method

The pseudocode explained in this section are separate pieces of important parts of the code, but not the entire pseudocode. The entire pseudocode is attached in Appendix A.

Process 1 and 2: The arrival and retrieval of containers

The first step in the algorithm determines where to place the containers at the terminal with the method devised in the previous section. The pseudocode for the arriving containers at the terminal is shown in Figure 14.

```
For time = MinDays To MaxDays 'Days that the application runs

    Do While current day = time 'Keep running till all containers this day are placed

        LocationContainer 'Call sub LocationContainer

        arrival = arrival + 1
        Remove incoming container from InputTable

    Loop 'Loop code for all incoming container current day
```

Figure 14: Pseudocode arriving containers

From Figure 14 the initiation of the algorithm can be seen. The first row of code determines the day in which the algorithm currently runs. And the following loop ensures that all containers for that day are placed at the terminal. The sub LocationContainer is a different sub which determines possible positions where the container can get stacked and determines if this position is most beneficial. The pseudocode of the LocationContainer sub is given in Figure 16.

After all the containers are placed at the bay, the retrieval of containers starts and with this retrieving the relocating. The pseudocode in Figure 15 shows the process of retrieving a container. As the pseudocode is quite repetitive for each row and to increaser the readability some parts of code are replaced with dots. However, the code that is supposed to be at the dots resembles the If statement underneath the *"If RowCounter = 2 Then"* part.

```
For RowCounter = 1 To 5 'Bay row counter
    For y = 1 To 6 'Bay width counter
        For x = 6 To 1 'Stack hight counter
            If container needs to be retrieved (std = 0) Then
                If RowCounter = 2 Then 'retrieve container from row 2
                    If there is a blocking container in rows before and after row 2 Then
                        For RelocationNr = Containers that need to be relocated To 0
                        'relocate containers till height stack 1 lower than height retrieving container

                            RowCounter = RowCounter - 1 'Relocate container from row 1
                                RelocateContainers 'Call sub RelocateContainers
                                relocation = relocation + 1
                            RowCounter = RowCounter + 1 'Restore rowcounter

                        Next RelocationNr
                    End If

                ElseIf RowCounter = 3 Then 'retrieve container from row 3
                    .....

                ElseIf RowCounter = 4 Then 'retrieve container from row 4
                    .....

                End If

                If container that needs to be retrieved <> top container stack Then
                   ..... 'relocate containers stacked on top of container which needs to be retrieved

                    Remove container from bay
                    retrieval = retrieval + 1

                Else 'retrieval with no relocation containers on top
                    Remove container from bay
                    retrieval = retrieval + 1
                End If
            End If
        Next x
    Next y
Next RowCounter
```

Figure 15: Pseudocode for relocating containers

The pseudocode in Figure 15 loops over the entire bay searching for containers that need to be retrieved on this day (containers with Etd = 0). Then, if the container that needs to be retrieved is stacked in row 2,3,4 it checks whether the container stack is accessible for the reach stacker (if there are blocking containers in the rows before and after). If it appears the case that there are blocking containers, then these containers are relocated. Next it is checked whether the container that needs to be retrieved is on top of the stack. If this is not the case, then the containers standing on top are also relocated. The relocating of containers happens in the RelocateContainers() pseudocode shown in Figure 16. After all the blocking containers are relocated, the container is retrieved, adding 1 to the retrieval counter.

Now for the pseudocode that determines the available locations for locating and relocating the containers, and the placement on these locations. As the pseudocode displayed in Figure 16 has some repetitive parts one example of this part is given and the others replaced with dots in order to make the code more readable. The If statement below the *"If CurrentRow = 2 Then"* line resembles the code that has been replaced with dots. In order to increase the readability, the codes for the relocating and locating of containers has been made into one sub. This is possible because when written in pseudocode the process of locating or relocating is the same, opposed to the Excel implemented algorithm where different restrictions and the placement of containers with a truck move were preventing the use of the same code.

```
Sub RelocateContainers() and LocationContainer()
'Find possible (re)location locations
    For CurrentRow = 5 To 1 'Loop over rows bay
        For j = 1 To 6 'Loop over width bay
            i = 6 'Loop over stack height
            Do While IsEmpty(Location above container) And j <> y
            'Loop over stack till top container found, and cannot relocate in same width as retrieving container (y <> j)
                If CurrentRow = 2 Then '(Re)locate to row 2
                    If top container found and container can get (re)stacked here under stacking restrictions Then

                        Save container location same Std
                        If Std > MaxStd Then
                            Std = MaxStd
                            Save location container MaxStd
                        End If


                    ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then

                        Save location ground level

                    End If

                ElseIf CurrentRow = 3 Then '(Re)locate to row 3
                    ....

                ElseIf CurrentRow = 4 Then '(Re)locate to row 4
                    ....

                Else '(Re)locate to row 1, 5
                    ....

                End if
            i = i - 1
            Loop
        Next j
        If location has been found Then Exit For 'If location in row found leave row loop
    Next CurrentRow
```

Figure 16: Pseudocode finding (re)location positions

The pseudocode in Figure 16 loops over the entire bay looking for spots to (re)locate the containers. The code saves the location of a container with the same Etd as the container that is to be (re)located, the location of the container with the latest leave date (MaxEtd), and the location of a possible ground level. When a location is found for a container in the current row then the CurrentRow loop is ended. As the container can get stacked in this row. Before the location of a container is saved, it is checked whether the container can get stacked here considering the stacking constraints of the reach stacker. If it appears the case that the container cannot get stacked in the current position, no location is saved, and the loop continues.

In Figure 17 the pseudocode for the placement of containers at the bay is shown. This pseudocode is the second part of the RelocateContainers() and LocationContainers() sub.

```
    'Placement (re)location containers
    If container same Std found Then

        (Re)locate container at spot same Std

    ElseIf spot on ground level found Then

        (Re)locate container on ground level

    ElseIf spot on container MaxStd found Then

        (Re)locate container at spot MaxStd

    ElseIf no spot found Then

        Add 1 to counter containers that cannot be (re)located

    End If
End Sub
```

Figure 17: Pseudocode (re)locating of containers

The pseudocode in Figure 17 places the containers at the terminal at the possible locations found by the pseudocode in Figure 16. Firstly, if a container location has been found with the same Etd as the (re)locating container, the container is stacked on top of this container. When no container location with the same Etd has been found the container is placed on the ground level, when possible. Thirdly, if no other location has been found, the container is stacked on top of the container with the highest Etd. If no places are available at the bay the container is not (re)located, and the counter for containers with no location or no relocation spot found increases by one.

Special case: The retrieval and return of containers by truck

As mentioned before in the "Input data" section the data sets that run through the algorithm consists of containers which do not all have a truck move. Thus, when placing a container at the terminal that has a truck transportation action in between their arrival and return by barge then these containers need to run through the algorithm twice. This is ensured at arrival to the bay, when a container with a truck move at processing by the algorithm inserts a new row in the "InputData" table, and the relevant container information copied, ensuring that the container runs through the system twice. This is done by calling the pseudocode from Figure 18.

```
Sub InsertInputRow()

d = Range("InputData").Rows.Count 'Range is the table "InputData"

For i = 1 To d 'Loop over containers yet to be placed
    If estimated arrival day container = estimated day truck move Then

        Insert input row in "InputData" at corresponding day
            'Fill empty row with relevant information
            copy B.ID of container
            arrival day container = days until truck transportation + time
            Std container = estimated days between truck movement and barge back
            copy carrier of container
            copy and invert empty status 'During truck move container is loaded / unloaded
        If Cell has been inserted Then Exit For 'If cell is inserted, leave for loop
    End If
Next i

End Sub
```

Figure 18: Pseudocode insert truck arrival

The pseudocode in Figure 18, starts with setting a range that incorporates the entire table "InputData". The code then loops over this range until the day that the truck is expected to return from its truck

movement. Once it finds the corresponding day, a row is inserted at this day and relevant container information copied into this new row. The estimated days between truck delivery and barge back becomes the Etd of this container and the container changes its empty status, as during a truck move the container is loaded or unloaded. Next, if the cell has been inserted and relevant information copied the for loop is left.

Random stacking method

In the random stacking method, the location in which the container is (re)located is randomized. This stacking method is developed to provide a baseline for the results of the other test to ensure their validity. The pseudocode for determining where to randomly stack the container is shown in Figure 19.

```
If top container found and container can get (re)stacked here under stacking restrictions Then

    RndNr = Rnd()
    If RndNr > 0.833 Then '1/6th chance of stacking container in available position
        Save location cell
    End If

ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then

    RndNr = Rnd()
    If RndNr > 0.833 Then
        Save location cell
    End If
End If
```

Figure 19: Pseudocode random stacking

The pseudocode in Figure 19 shows the way a random stacking location is determined. This code replaces the code that determines the possible container locations from the new stacking method in Figure 16. The chance of placing the container at the random position is 1/6 as there are 6 stacking locations in a row. The code loops until a random stacking position has been found under the stacking restrictions posed by the reach stacker or the loop is left when there are no more available stacking positions at the bay.

Current stacking method

Currently, the containers are being stacked corresponding to their carriers, to simulate this a stacking method is developed which places the containers according to this carrier. The names of the carriers have been anonymised and replaced with numbers to ensure the data confidentiality. The pseudocode is shown in Figure 20.

```
If top container found and container can get (re)stacked here under stacking restrictions Then

    LastCarrier = carrier of container top stack 'Save carrier container top stack

    If Carrier = LastCarrier Then 'If carrier (re)locating container = carrier container top stack
        Save location cell same carrier
    End If

    If Carrier <> LastCarrier Then
        Save location cell other carrier
    End If

ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then
    Save location cell ground level
End If
```

Figure 20: Pseudocode carrier stacking

Figure 20 displays in the same manner as in Figure 19 the way the possible locations for the containers are determined. This code replaces the code which determines the possible container locations from the new stacking method in Figure 16. In the carrier stacking method, the containers are firstly placed on top of a container that has the same carrier. If no corresponding carrier has been found the container gets stacked on the ground level of the bay. In the case that no ground levels are available anymore, then the container is (re)located on top of a container of another carrier.

Yard crane implementation

In order to expand the scope of this research, and to widen the options for implementation at other terminals the performance of the implementation of a yard crane is examined. The benefits of the implementation of a yard crane is that it greatly reduces the stacking restrictions. Reach stackers are bound to restrictions previously mentioned which limit them in their stacking capabilities, opposed to yard cranes which can reach every top stack in the bay. Pictures and explanation of these two different stacking machines were given in Chapter 1 Section 1.5.

The adaption of the algorithm to make it correspond to the stacking method of a yard crane is simple. Two instances are tested, the implementation of a yard crane using the current and new stacking method. This is the same method of stacking as displayed in Figures 14, 15, 16, 17 and 20. With the only difference being the stacking restrictions of the reach stacker that no longer apply to the yard crane. Recall that the stacking restrictions of the reach stacker are that it cannot stack further than 2 rows. So, when removing these stacking restrictions posed by the reach stacker an algorithm assessing the performance of a yard crane is implemented.

Validity

The validity of the implemented algorithms will mainly follow from the numerical results that they will produce, but the assumptions made, and the structure of the algorithm also influence the validity of the results. In this section these points are discussed, and the impact of them discussed.

All containers arrive before departing in the algorithm. Firstly, all containers that arrive on a day are placed at their designated spots of the bay, before the retrieval of containers start. In the real life situation not all arriving containers arrive, before retrieval starts. But, as the algorithm first places the containers before removing them, the number of relocation moves increase, as more containers are placed in the bay before retrieval starts.

All containers are scheduled on a day not a timeslot. As mentioned before, all containers arrive and leave the terminal on the same day. But when multiple barges arrive during a day then containers that leave on an earlier barge need to leave before the containers scheduled on a later barge and there could also be possible placement of containers in between barge arrivals. The situation at the terminal is that usually not more than one or two barges a day visit, so in a situation where little barges arrive and leave the algorithm is more accurate than in situations where there is more throughput. This issue can get fixed by planning containers on a timeslot rather than on a day.

The algorithm is tested with the same size containers. It is assumed that containers of different sizes get stacked in different bays. This does not heavily influence the validity of the algorithm, but possible performance could improve when containers of different sizes get stacked within the same bay.

The algorithm is run with the historical arrival and departure dates. In the algorithm the uncertainty in container leave date is not considered and of all containers it is known beforehand when they need to leave again. In the real life situation this is not always the case. For half of the containers that arrive by barge it is known when they leave by barge again. And of a negligible 2% of the containers with a truck move it is known when they need to leave the terminal again. These containers, with no estimated leave date, can receive an estimation of their own by means of the historical data. In Section 3.2 Figure 9 these average times are displayed. These can be given to the containers when their estimated leave date is unknown.

The above points are the differences between the stacking method in the algorithm and the stacking environment in a real life situation. As the algorithm not entirely accurately reflects the real life situation, the performance may differ. But, when differences in performance differ greatly, and the newly devised stacking method outperforms the current stacking policy, it can be said that this new method of stacking outperform the current policy.

4.3 CONCLUSION

In this chapter a new method for containers stacking is described and implemented into VBA. Two main research question have been answered. Firstly, *"How will the new stacking algorithm be shaped?"*. In Section 4.1 the new stacking method is explained, together with the other stacking policies which will serve as a baseline for the validity of the algorithm and to compare the performance of the newly devised method to the others.

In Section 4.2 the pseudocode of the different stacking policies and methods are displayed. This pseudocode has been implemented into VBA to run the code and get the results discussed in Chapter 5. The research question answered in this chapter is: *"Is the implemented method of stacking valid?"*. The validity of the proposed stacking method is mentioned in the second part of Section 4.2. There are a few differences between the stacking environment assumed by the algorithm and the real life situation. However, all the different stacking methods and policies in the simulated environment run under the same restrictions. Thus, the performance of the current stacking policy can be compared to the performance of the new stacking method as they run under the same restrictions. Following these consequences, the results of the algorithm will differ from the performance in the real life situation. But when comparing the results between the different stacking methods and policies the performance can be assessed.

5. EVALUATION OF RESULTS

In Chapter 5 the results of the implemented algorithm of Chapter 4 are discussed. The main research question of this chapter is: *"What performance can be expected when implementing the different methods?"*. Subdivided into the following two sub-research questions: "*What is the performance of the different stacking methods and are the results valid?"* answered in Section 5.1, and *"What is the quality of the generated solution?"* answered in Section 5.2.

5.1 EXPERIMENT RESULTS

In the section the experimental setup is explained. After which the results from these experiments will be given and compared to each other.

Experimental setup

Recall from Section 4.2 that the input data used for the experiments are 3 datasets consisting of 1000, 2000, and 3000 input containers respectively, randomly retrieved from the dataset of 3052 containers from Cofano. These containers are placed at and retrieved from the terminal over a 156 day span. These 1000, 2000, and 3000 containers of input data respectively give 5227, 3461, and 1709 arriving containers because the containers which have a truck retrieval in between their arrival and leave by barge run through the system twice.

All different stacking methods and policies are run with these three different database sizes. The random stacking policy is run 5 times per dataset respectively because of the randomness of the algorithm. In the other stacking algorithms there is no randomness, thus only one run of these algorithms is necessary. A visual representation of the bay during stacking is displayed in Appendix B.

Results

The complete results from the experiments can be found in Appendix C. In this section the results from the experiments will be compared to each other to assess their performance with the main performance indicator being the percentage of relocation moves performed at the terminal. In all cases the computer used in the computational experiments was equipped with an Intel Core i5-3470 CPU 2.90 GHz and 16 GB of ram

Random stacking method

Recall that in the random stacking method the containers are randomly placed at the terminal under the reach stacker stacking constraints. A selection of the more relevant results from these experiments is displayed in Table 8. The results on each dataset are the averages of 5 runs.

Random (avg)	**Relocation**	**Avg. capacity**	**max capacity**	**Nr. No location**	**Nr. No relocation**
Big (3000)	34.57%	44.85%	66.46%	747	155
Medium (2000)	33.80%	35.17%	63.56%	139	53
Small (1000)	28.15%	18.65%	41.70%	0	0
Average	*32.17%*	*32.89%*	*57.24%*	*295.33*	*69.33*

Table 8: Selection results random stacking method

From Table 8 it can be seen that the random stacking method has a high relocation percentage that does not significantly reduce with smaller data sets. This can be explained by the fact that the random

stacking method fills up the 1st and 5th row quicker than in the other methods, as the other methods systematically fill the bay from the 5th to the 1st row. When containers are located in the 1st and 5th row they often need to be relocated to reach the containers that are stacked in between. The quick filling of the 1st and 5th row also causes the high number of un(re)locatable containers in the bigger two datasets. When the 1st and 5th row start to fill up containers can then only be stacked in these rows and once these rows are full no location for the container can be found. This does not occur in the small data set, as not enough containers run through the system to fill up both the 1st and 5th row.

Current stacking method

Recall that in the current stacking method the containers are placed on top of containers with the same carrier under the stacking restrictions posed by the reach stacker. A selection of the results of these experiments can be seen in Table 9.

Current	Relocation	Avg. capacity	max capacity	Nr. No location	Nr. No relocation
Big (3000)	34.18%	51.01%	79.00%	380	76
Medium (2000)	33.66%	37.62%	73.00%	20	7
Small (1000)	27.75%	18.65%	42.00%	0	0
Average	*31.86%*	*35.76%*	*64.67%*	*133.33*	*27.67*

Table 9: Selection results current stacking method

From Table 9 it can be seen that in the current stacking method there is a bigger relocation percentage compared to the relocation percentage following from the historical data of the terminal. The real life terminal now achieves a relocation percentage of 21.16%, which is lower than the percentages of all the experiments. This can however be explained by the fact that in the experiments more containers are processed through the bay compared to the real life situation. In total 18362 containers are processed by the terminal in real life over the same time span. These 18362 containers spread over the 25 bays of the terminal result in 734 containers per bay per year. As the algorithm runs 5227, 3461, and 1709 containers respectively through the bay a higher percentage was to be expected.

New stacking method

The algorithm of the new stacking method places the containers at the terminal according to the method explained in Section 4.1. A selection of the results can be found in Table 10.

New	Relocation	Avg. capacity	max capacity	Nr. No location	Nr. No relocation
Big (3000)	19.26%	51.04%	86.00%	339	56
Medium (2000)	11.45%	37.30%	73.00%	34	8
Small (1000)	3.42%	18.65%	42.00%	0	0
Average	*11.37%*	*35.66%*	*67.00%*	*124.33*	*21.33*

Table 10: Selection results new stacking method

From Table 10 it can be seen that the relocation percentage greatly reduces, decreasing with each smaller data set. This can be explained by the fact that with more containers processed through the bay the less ideal positions (containers same Etd) can be found. Thus, when placing these containers at the less ideal positions the relocation percentage increases. Compared to the real life situation the relocation percentage is also reduced. In the real life situation, a relocation percentage of 21.16 is

achieved with 734 containers run through the bay, so the new stacking method outperforms the real life situation. However, in the real life situation the terminal may choose to place more containers in bays closer to the barges to reduce transportation time. So, the number of containers processed per bay may not be entirely accurate. But the reduction in relocation percentage is so significant that it can be said with certainty that the new stacking method outperforms the real life stacking method under the assumptions of the algorithm. With the new stacking method there are still some containers that cannot be located or relocated at the terminal. This is the case because in the large database many containers are run through the bay reaching almost maximum capacity.

Yard crane implementation

The possibility of implementing a yard crane is also assessed as option. The implementation of a yard crane reduces the relocation percentage as the stacking restrictions posed by the reach stacker no longer apply. In order to show the improvement of the new stacking method compared to the current stacking policy the yard crane option has been examined with both algorithms. A selection of the results of the implementation of a yard crane system with the current stacking policy can be found in Table 11.

YC (current)	Relocation	Avg. capacity	max capacity	Nr. No location	Nr. No relocation
Big (3000)	26.40%	54.36%	94.00%	176	0
Medium (2000)	26.48%	37.98%	79.00%	0	0
Small (1000)	26.50%	18.65%	42.00%	0	0
Average	*26.46%*	*37.00%*	*71.67%*	*58.67*	*0*

Table 11: Selection results yard crane implementation current stacking method

Table 11 shows the results of the implementation of a yard crane and the usage of the current stacking policy. It can be seen that the relocation percentage is almost exactly the same in all test cases. The reason for this being that almost all containers can be stacked on top of containers which have the same carrier. But these containers are still stacked randomly considering their Etd as the current stacking method algorithm does not take Etd's into account, resulting in relocation moves.

YC (new)	Relocation	Avg. capacity	max capacity	Nr. No location	Nr. No relocation
Big (3000)	0.61%	56.79%	100.00%	40	0
Medium (2000)	0.00%	37.98%	79.00%	0	0
Small (1000)	0.00%	18.65%	42.00%	0	0
Average	*0.20%*	*37.81%*	*73.67%*	*13.33*	*0*

Table 12: Selection results yard crane implementation new stacking method

Table 12 shows that when implementing a yard crane with the new stacking method the relocation percentage jumps to 0. This is the case because containers are scheduled to leave on a day, and when implementing the yard crane almost all (except for a few containers in the big data set) are stacked on top of containers with the same Etd or a higher Etd. This is possible because the yard crane can reach all top containers of the stacks. The implementation of a yard crane clearly is the best option when wanting to reduce the relocation percentage. However, such low percentages are only possible because there is little throughput at this small inland terminal. Should throughput increase, an increase

in relocation percentage can be expected. Furthermore, the costs of the implementing a yard crane may not outweigh the benefits as it is an expensive system.

It can be seen that the new stacking method greatly increases the performance of a yard crane implementation compared to the current stacking policy. With the cause for this being the stacking taking into account the Etd of containers.

Comparison

Now the performance of the different methods will be compared to each other. The relocation percentages of all the experiments are shown in Table 13 and visualised in Figure 21.

Data set	Relocation Rnd	Relocation Cur	Relocation New	Relocation YC (current)	Relocation YC (new)
Big (3000)	34.57%	34.18%	19.26%	26.40%	0.61%
Medium (2000)	33.80%	33.66%	11.45%	26.48%	0.00%
Small (1000)	28.15%	27.75%	3.42%	26.50%	0.00%
Average	*32.17%*	*31.86%*	*11.37%*	*26.46%*	*0.20%*

Table 13: Comparison relocation percentage different methods

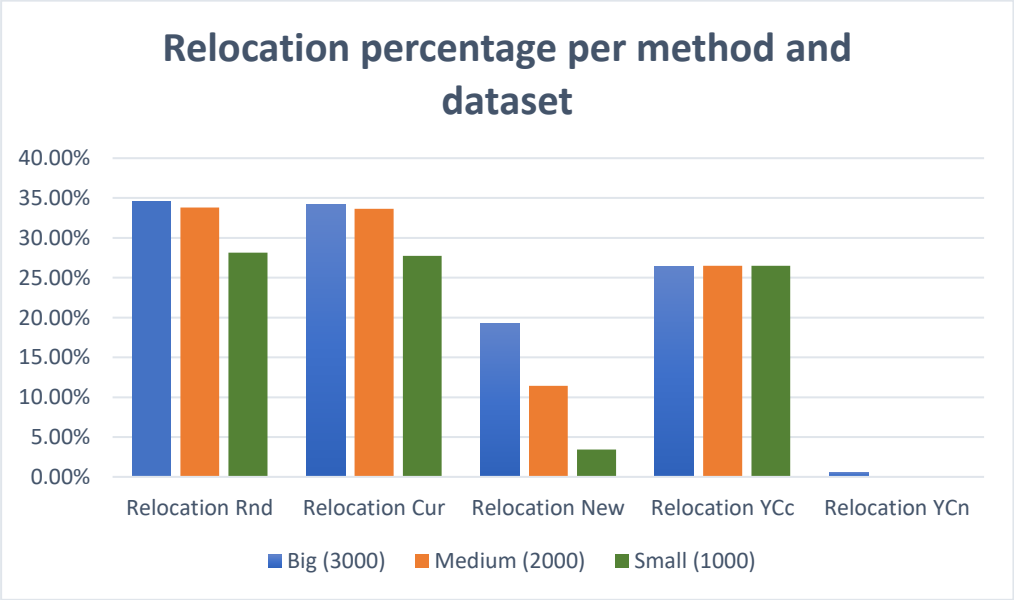

Figure 21: Relocation percentage per stacking method and dataset

When looking at the differences in relocation percentage displayed in Table 13 and Figure 21 it can be seen that the random stacking method has almost the same performance as the current stacking method. This can be explained by the fact that when stacking the containers per carrier without any attention towards the Etd of containers, the containers are still randomly stacked considering their Etd. Also, in the real life situation the containers of different carriers are spread over multiple bays whilst in the algorithm the containers of all carriers are stacked within the same bay. When spreading the containers of carriers over multiple bays, more space is created to stack containers according to their carrier, whilst in the algorithm often containers of the same carrier could not be stacked on top of each other. When containers are placed in clumps of just the same carrier the relocation percentage can be reduced as the average Etd of containers vary per carrier. This can be seen in Section 3.2 Figure

9. However, in the real life situation the expected leave date is not considered, so the stacking per carrier still occurs quite randomly when looking at their Etd. The current stacking method does outperform the random stacking method taking into account the containers that cannot be located or relocated at the terminal, as can be seen in Table 8 and 9. This meaning, that the current stacking method has a more efficient space usage compared to the random stacking method.

Furthermore, in Figure 21 it can be seen that the new stacking method outperforms the current stacking method with a significant decrease in relocation percentage. When running the big data set the performance of the algorithm comes near the performance of the real life situation with a relocation percentage of 19.26 compared to the 21.17 in the real life situation. However, the big dataset runs 6 times more containers through the terminal compared to the real life situation, significantly increasing the capacity of the terminal. The smaller data sets achieve an even lower relocation percentage, 11.45 and 3.42 respectively. The goal of reducing the relocation percentage at the terminal to 13.94% is thus reached. Even when running the medium dataset, which is almost 3 times larger than the number of containers currently being processed by the terminal per bay, the relocation percentage is 11.45%.

The acquisition of a yard crane reduces the relocation percentage even further, achieving 0% relocation moves with the two smaller data sets and 0.61% with the big dataset as can be seen in Table 13. This is however just when the new stacking method is used. When implementing a yard crane using the old stacking policy there is no significant improvement compared to the new stacking method, and no significant reduction in relocation percentage is achieved. The acquisition of a yard crane combined with the new stacking method is the most ideal solution when aiming to reduce the relocation percentage. They are however expensive to acquire, and the costs may not outweigh the benefits of implementing such a system, especially for smaller terminals with less container movement and relatively more free space when comparing to larger terminals.

Validity

The validity of the algorithm itself is already discussed at the end of Section 4.2, but the validity of the results is not included in this section. The results from the experiments are very positive, as the number of relocation moves are greatly reduced. The results from the experiments them self are valid, as there are no illogic results. The random algorithm performs the worst together with the carrier stacking method. That these two method performances are so close can be explained by the fact that in the carrier stacking method the containers are still randomly stacked according to their Etd, which causes a lot of relocations. The current stacking method does outperform the random stacking method when looking at the containers that cannot be located, as the stacking occurs more systematically from back to front.

The results of the new stacking method appear valid. They appear to not be unreasonably low but do outperform the current stacking method. The results of the small database are almost optimal (relocation percentage nearing to 0), and this can be explained by the fact that the terminal has a lot of free space when running the smallest database. The average capacity of the terminal is 18.65% and the max capacity 42% when running the small database with the new stacking method as can be seen in Table 10. But also, when running the two bigger datasets and free space at the terminal decreases, still a significant decrease in relocation can be seen.

The results from the yard crane may appear to be unrealistically low. But, under the choices of the test instances they are quite logical. The yard crane can almost always place containers on the container with the same Etd. And when there are only containers that are stacked on containers with the same Etd they never need to be relocated, as a yard crane can reach the top of all the container stacks.

When reviewing the results from the algorithm they all appear to be valid under the assumptions and choices made. There are no weird results that are illogic when comparing them to the other results.

5.2 QUALITY OF THE GENERATED SOLUTION

In this section the quality of the generated solution will be assessed. The results from the experiments are positive and show a significant decrease in relocation percentage. The results appear to be valid under the assumptions made (Section 4.2) and Scope (Section 1.5). The assumptions however cause the algorithm to differ from the real life situation. Especially the assumptions taking into account the *arrival and leave date* that often is unknown beforehand and must be given to the containers by an estimation based on their carrier. And the *scheduling of containers on a day* are assumptions that might influence the results of the experiments. When all containers receive an Etd, and when this estimation can be accurately made, the algorithm can be implemented into the real life situation.

As mentioned previously in the validity section of Section 4.2 the container information that was used in the experiments was full complete information of the arrival and leave of containers. In the real life situation however, this is unknown for a large part of the containers. These containers can receive an estimated leave date based on the historic data of the time between arrival and leave of the containers of different carriers as shown in Section 3.2 Figure 9, but this new estimation is not used in the experiments. Running the algorithm with this estimation will certainly alter the results of the experiments and increase relocation percentage, as there is more uncertainty in the leave date of containers. However, the improvement in relocation percentage in that large that an increase in performance can still be expected.

Furthermore, all containers are scheduled to arrive and leave in the timespan of a day. This is accurate if there is only one barge arriving and leaving each day, this is often not the case at bigger terminals. The terminal of whom the data was gathered, is a small inland terminal where often one or two barges arrive a day. Considering this the algorithm approaches reality well for small inland terminals. However, when running the algorithm with the data of terminals where more barges arrive it might be better to schedule containers on the timeslot at rather than on a day to ensure the validity of the algorithm.

5.3 CONCLUSION

In this chapter we wanted to answer the research question "*What is the performance of the different stacking methods and are the results valid?*". The performance of the algorithm satisfies the goal set at the beginning of the report. The new stacking method outperforms the current stacking method and achieves a bigger improvement than the goal set at the beginning of the report. Also, a yard crane implementation greatly reduces the relocation percentage, when implementing it together with the new stacking method. The results from the experiments are valid, as there are no illogical values and all factors which might seem illogical can be explained (such as the random and current method results being so alike).

The second research question answered in this chapter is "*What is the quality of the generated solution?*" The results of the experiments have some validity issues. Mentioned in Section 5.2. These issues will influence the performance of the algorithm in real life but are not that influential that the current stacking method will outperform the new stacking method. Thus, the algorithm does not fully reflect the real life situation, but the results from the experiments are well enough that a decrease in relocation percentage can be expected when implementing the new stacking method. With the main quality issue being the assumption that all containers have an Etd at arrival to the terminal. But once a way has been found to give all arriving containers this Etd the algorithm can be implemented in practice.

6. CONCLUSIONS, RECOMMENDATIONS AND LIMITATIONS

In this chapter the conclusions of the research are given in Section 6.1. Furthermore, recommendations for the company and future research are given in Section 6.2. Lastly, a discussion about the limitations of the algorithm is included in Section 6.3.

6.1 CONCLUSIONS

The core problem that is addressed in this research is that "*There is no computed method behind the current policy of container stacking.*" This absence of a stacking method resulted in higher than necessary relocation percentages at the terminal. In order to decrease this, a new stacking method has been devised and programmed into an algorithm to assess its performance by means of numerical analysis. The main research questions from each chapter will now be discussed.

In Chapter 2 the following sub question is answered: *"What are possible solutions to the container stacking problem that can be found in the literature?"*. Multiple approaches for the stacking problem were found, most of them taking into account the relocation percentage. The knowledge from the literature was used to select the appropriate approach, the Loading and Unloading problem discussed by (Lehnfeld & Knust, 2014) and to give some background to the stacking problem that helped avoid pitfalls. By the devising of the new stacking method the greedy stacking heuristic of (Gharehgozli, Roy, & de Koster, 2016) was used as initial method, and further refined. Furthermore, the literature gave some options for necessary generalisations in the algorithm.

In Chapter 3 discusses the research question: *"How can the solution approach be shaped, incorporating the available data?"*. In this chapter the available data is analysed, and a new stacking approach found. The data gives insights in the estimated time containers stay at the terminal and provides insights in terminal container handling. The solution approach is determined based on the conclusions from the data analysis and with help from the literature. The approach that is chosen is the stacking of containers considering their (given) estimated leave date.

In Chapter 4 the following research question is addressed: *"How should the stacking algorithm work?"*. Firstly, in Chapter 4 a new stacking method is devised and explained stacking the containers according to their Etd. The algorithm firstly places containers on top of containers with the same Etd, secondly on a ground level, and thirdly on top of an already stacked containers with the highest Etd. This stacking method is a refinement of the greedy stacking heuristic proposed by (Gharehgozli, Roy, & de Koster, 2016). Furthermore, a random and current stacking method are set up to make the results from the tests comparable. Also, the possible implementation of a yard crane is examined. To assess the performance of the yard crane the new and current stacking method were considered together as stacking methods with this implementation. In Section 4.2 the pseudocode of the implementation of the algorithm in VBA is explained.

Chapter 5 discusses the results of this research with the following research question: *"What performance can be expected when implementing the different methods?"*. From the results it can be seen that the new stacking method outperforms the current stacking policy. The relocation percentages are reduced whilst running about 2/3/6 times more containers through the bay than in the current situation (respectively per dataset). Thus, an improved stacking method has been devised and is probable to outperform the current stacking method in a real life situation. The implementation

of a yard crane further reduces the relocation percentage when implementing this with the new stacking method, but the trade-off between the costs and benefits of such a system has not been made.

Now the core problem "*There is no computed method behind the current policy of container stacking*" can be addressed. In this research a computed method for container stacking has been determined and according to the tests, outperforms the current method of containers stacking, reducing the number of relocation moves. The advice for implementing the new stacking method is given in the section recommendations (Section 6.2). Furthermore, the option for implementing a yard crane is assessed and proves to greatly reduce the number of relocation moves.

It cannot be said with certainty that the new stacking method will outperform the current stacking method as the test instances operates under assumptions that differ from the real life situation. With the main assumption being that all containers have an Etd at arrival to the terminal. The improvement of the new stacking method, compared to the current stacking method, is however that significant that a reduction of relocation moves, when implementing the new stacking method, is probable.

6.2 RECOMMENDATIONS

The recommendations are split into two different categories. Firstly, recommendations for Cofano will be discussed in the *company* subsection explaining the contribution to practice. Secondly, the recommendations for future research will be discussed in the second subchapter contributing to the theory in this field of research.

Company

The goal of Cofano is to develop an application where terminals can enter their arriving and leaving containers and determines where containers need to be placed or relocated. The devised algorithm with the new stacking method is a first step in the development of such an application. The improvement in performance of the new stacking method is of such significance that a real world improvement is probable, so Cofano is recommended to adapt / expand the new stacking method and with this algorithm build their desired application. The algorithm itself is not yet ready to be implemented into a real time application as there are assumptions made that do not correspond with the real life situation. The containers that arrive to the terminal all need to receive an Etd before they can be placed. Cofano is recommended to explore options for assigning containers said Etd, a method has been proposed in this research, but the possible performance of it is not analysed. The results of the algorithm can however be used to show that the new stacking method will most probable outperform the old one and show the possible improvement that can be made.

In the current algorithm it is assumed that containers of different sizes get stacked in different bays. Possibly when stacking containers of different sizes in the same bay terminals might be more persuaded to incorporate the new stacking method, as it gives a wider variety of options. So, the possibility to incorporate this can be examined.

The algorithm can be improved to further reduce the relocation percentage. Options for further improvement of the algorithm are given in the *Future research* section. The recommendation is to first look at possible improvements Cofano can make to the algorithm and then develop the desired application.

Future research

The contribution to the field of literature is a piece of research with practical application into the Loading and Unloading (LU) problem. That, as of yet, has not been widely addressed in the field of research as a combined problem. A more refined stacking heuristic has been proposed expanding on the greedy stacking heuristic of (Gharehgozli, Roy, & de Koster, 2016), where containers are always firstly placed on ground level. Opposed to the stacking method proposed in this research, where containers are firstly stacked on top of containers with the same Etd.

The recommendations for future research provide options where the algorithm can still be improved. Firstly, a non-event based method can improve the quality of the algorithm. In the current algorithm container moves are performed just when containers arrive and leave the system. In a non-event based approach containers that need to be relocated for the retrieval of a container in the future are already relocated during idle time, reducing the number of unwanted relocations in the future when time is of the essence.

Furthermore, in the new stacking method when placing a container on top of a container with the MaxEtd just the top containers are considered. Whilst it is possible that an underlying container needs to leave earlier, causing more relocation moves when retrieving said underlying container. So, considering the entire stack of containers when placing containers on MaxEtd instead of only the top containers reduces the number of relocation moves further. However, as not that many containers are stacked according to their MaxEtd the decrease will be marginal.

When incorporating the empty or full status of containers into the algorithm possible improvement is possible. The restrictions for stacking containers with the reach stacker differ for empty and non-empty containers. Recall that non-empty containers can be stacked over 1 row if the back row is stacked 1 higher than the front. For empty containers this rule extends one row. It is possible for a reach stacker to retrieve or locate empty containers over 2 rows, if both rows are 2 and 1 container lower respectively. This gives more locating options than the current algorithm that does not take this into account.

6.3 LIMITATIONS

In every algorithm / simulation / model limitations exist, these limitations will be discussed in this section.

The algorithm runs with the historical data of the containers. This meaning that it runs under the assumption that the arrival and leave date of the containers is known beforehand. In the real life situation of only 50% of the containers the leave date of barge transports is known and just 2% of truck transports. The containers that do not have an Etd can receive one based on the average time a container stays at the terminal per carrier. These estimates come from the historical data and are shown in Section 3.2, Figure 9. Should this historical data prove to be too inaccurate then the containers can receive a timeframe rather than a date on which they need to leave. This assumption influences the performance of the algorithm in a real life situation compared to the simulated one. The improvement in relocation percentage is however that high, that the new stacking method will probably still outperform the current one.

In the new stacking method, the performance is only assessed with the usage of one bay. To get a better view on the performance over an entire terminal the algorithm can be adapted to run over multiple bays with more input containers, better replicating the real life situation.

The algorithm assigns the containers a day on which they need to arrive and depart. However, when multiple barges arrive and leave during a day then this timeframe is inaccurate as containers that need to arrive / leave on the first barge need to be located / retrieved before the containers of later barges. In the current model the current situation is quite accurate as usually 1 / 2 barges arrive each day. But when implementing the algorithm at other larger terminals it might be better to assign the Etd's on timeframes rather than days.

In the algorithm all containers arrive at the terminal before the retrieval starts. In the real life situation this is not always the case as containers can also be retrieved first. Especially when multiple barges arrive each day. However, as in the algorithm all containers are first placed before retrieval starts the algorithm will perform better in the real life situation. This because the bay will be emptier when containers are retrieved before placement starts reducing the number of relocation moves.

The proposed algorithm does not consider areas designated for dangerous goods. Certain containers cannot be placed next to others because of safety protocols. These types of containers are not considered by the algorithm. This does however not influence the performance of the algorithm, as these containers are placed in a different area than the bay the algorithm considers.

REFERENCES

Dekker, R., Voogd, P., & Asperen, E. v. (2006). Advanced methods for container stacking. -, 1-24. doi:10.1007/s00291-006-0038-3

Expósito-Izquierdo, C., Lalla-Ruiz, E., Armas, J. d., Melián-Batista, B., & Moreno-Vega, J. M. (2015). A heuristic algorithm based on an improvement strategy to exploit idle. *Elsevier*, 410-424. Opgehaald van https://doi.org/10.1016/j.cie.2015.05.030

Gambardella, L. M., Bontempi, G., Taillard, E., Romanengo, D., Raso, G., & Piermari, P. (1996). Simulation and Forecasting in Intermodal Container Terminal.

Gharehgozli, A., Roy, D., & de Koster, R. (2016). Sea container terminals: New technologies and OR models. *Maritime Economics & Logistics*, 103-140. doi:https://doi-org.ezproxy2.utwente.nl/10.1057/mel.2015.3

Hartmann, S. (2004). Generating scenarios for simulation and optimization of container terminal logistics. *OR Spectrum*, 171-192. doi:10.1007/s00291-003-0150-6

Kefi, M. K. (2007). Heuristic-based model for container stacking problem. *International Conference on Production Research-ICPR (Vol. 7)*.

Lehnfeld, J., & Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey. *European Journal of Operational Research*, 297-312.

López-Plata, I., Expósito-Izquierdo, C., Lalla-Ruiz, E., & Melián-Batista, B. (2016). Minimizing the Waiting Times of block retrieval operations in stacking. *Elsevier*, 70 - 84. Opgehaald van https://doi.org/10.1016/j.cie.2016.11.015

Lozancic, A. (2017, 01 20). *White Box Testing and Black Box Testing*. Opgehaald van gauss development: https://gauss-development.com/white-box-testing-black-box-testing/

Rei, R. J., & Pedroso, J. P. (2011). Heuristic search for the stacking problem. *International Transactions in Operational Research*, 1-17. doi: 10.1111/j.1475-3995.2011.00831.x

Rendl, A. &. (2013). Constraint models for the container pre-marshaling problem. *ModRef, 2013, 12th*, 1-13.

Timmer, R. (2012). *Container stacking and retrieval strategies for problems with full information.*

APPENDIX A: ENTIRE PSEUDOCODE NEW STACKING METHOD

Appendix A.1: Pseudocode Sub StackContainers()

```
Sub StackContainers()

For time = MinDays To MaxDays 'Days that the application runs

    Do While current day = time 'Keep running till all containers this day are placed

        LocationContainer 'Call sub LocationContainer

        arrival = arrival + 1
        Remove incoming container from InputTable

    Loop 'Loop code for all incoming container current day

    'Save avg capacity terminal
    avg capacitty = ((time - 1) * avg capacity + current capacity) / time 'Average capacity bay

    If current capacity > maximum capacity Then
        current capacity = maximum capacity
    End If

        'Retrieve containers Std = 0, and relocate blocking containers
    For RowCounter = 1 To 5 'Bay row counter
        For y = 1 To 6 'Bay width counter
            For x = 6 To 1 'Stack hight counter
                If container needs to be retrieved (std = 0) Then
                    If RowCounter = 2 Then 'retrieve container from row 2
                        If there is a blocking container in rows before and after row 2 Then
                            For RelocationNr = Containers that need to be relocated To 0
                            'relocate containers till height stack 1 lower than height retrieving container

                                RowCounter = RowCounter - 1 'Relocate container from row 1
                                    RelocateContainers 'Call sub RelocateContainers
                                    relocation = relocation + 1
                                RowCounter = RowCounter + 1 'Restore rowcounter

                            Next RelocationNr
                        End If

                    ElseIf RowCounter = 3 Then 'retrieve container from row 3
                        If there is a blocking container 2 rows before and after row 3 Then
                            For RelocationNr = Containers that need to be relocated To 0
                            'Containers 2 rows before or after need to be empty to reach 3rd row for retrieval

                                RowCounter = RowCounter - 2 'Relocate all container from row 1
                                    RelocateContainers
                                    relocation = relocation + 1
                                RowCounter = RowCounter + 2 'restore rowcounter

                            Next RelocationNr
                        End If

                        If there is a bocking container 1 row before and after row 3 Then
                            For RelocationNr = Containers that need to be relocated To 0
                            'relocate containers till height stack 1 lower than height retrieving container

                                RowCounter = RowCounter - 1 'Relocate container from row 2
                                    RelocateContainers
                                    relocation = relocation + 1
                                RowCounter = RowCounter + 1 'restore rowcounter

                            Next RelocationNr
                        End If
```

```
                ElseIf RowCounter = 4 Then
                    If there is a blocking container in rows before and after row 4 Then
                        For RelocationNr = containers that need to be relocated To 0
                        'relocate containers till height stack 1 lower than height retrieving container

                            RowCounter = RowCounter + 1
                            If relocating container = container with Std "0" Then
                            'As loop runs from row 1 to 5, possibility of relocating containers with std = "0"
                                remove container from bay
                                retrieval = retrieval + 1
                            Else 'If Std of container is not 0, relocate
                                RelocateContainers
                                relocation = relocation + 1
                            End If
                            RowCounter = RowCounter - 1

                        Next RelocationNr
                    End If
                End If

                If container that needs to be retrieved <> top container stack Then
                    For RelocationNr = Nr of containers on top retrieving container To 0

                        RelocateContainers
                        relocation = relocation + 1

                    Next RelocationNr

                    Remove container from bay
                    retrieval = retrieval + 1

                Else 'retrieval with no relocation containers on top
                    Remove container from bay
                    retrieval = retrieval + 1
                End If
            End If
        Next x
    Next y
Next RowCounter


'For each day take 1 of the Std of the containers
    For Row = 0 To 4
        For i = 2 To 7
            For j = 2 To 7
                Container Std = Container Std - 1
            Next j
        Next i
    Next Row

Next time

'Write counters in cells
arrivalcounter = arrival
retrievalcounter = retrieval
relocationcounter = relocation

End Sub
```

Figure 22: Pseudocode sub StackContainers()

Appendix A.2: Pseudocode subs (Re)LocateContainers()

```
Sub RelocateContainers() and LocationContainer()
'Find possible (re)location locations
    For CurrentRow = 5 To 1 'Loop over rows bay
        For j = 1 To 6 'Loop over width bay
            i = 6 'Loop over stack height
            Do While IsEmpty(Location above container) And j <> y
            'Keep looping over stack till top container found, and cannot relocate in same width as retrieving container (y <> j)
                If CurrentRow = 2 Then '(Re)locate to row 2
                    If top container found and container can get (re)stacked here under stacking restrictions Then

                        Save cell location same Std
                        If Std > MaxStd Then
                            Std = MaxStd
                            Save cell location MaxStd
                        End If

                    ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then

                        Save location ground level

                    End If

                ElseIf CurrentRow = 3 Then '(Re)locate to row 3
                    If top container found and container can get (re)stacked here under stacking restrictions Then

                        Save cell location same Std
                        If Std > MaxStd Then
                            Std = MaxStd
                            Save cell location MaxStd
                        End If

                    ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then

                        Save location ground level

                    End If

                ElseIf CurrentRow = 4 Then '(Re)locate to row 4
                    If top container found and container can get (re)stacked here under stacking restrictions Then

                        Save cell location same Std
                        If Std > MaxStd Then
                            Std = MaxStd
                            Save cell location MaxStd
                        End If

                    ElseIf groundlocation and container can get (re)stacked here under stacking restrictions Then

                        Save location ground level

                    End If

                Else '(Re)locate to row 1, 5
                    If top container found Then

                       Save cell location same Std
                        If Std > MaxStd Then
                            Std = MaxStd
                            Save cell location MaxStd
                        End If

                    ElseIf groundlocation and container can get restacked here under stacking restrictions Then

                        Save location ground level

                    End If
                End if
            i = i - 1
            Loop
        Next j
        If location has been found Exit For 'If location in row found leave row loop
    Next CurrentRow
```

```
    'Placement (re)location containers
    If container same Std found Then

        (Re)locate container at spot same Std

    ElseIf spot on ground level found Then

        (Re)locate container on ground level

    ElseIf spot on container MaxStd found Then

        (Re)locate container at spot MaxStd

    ElseIf no spot found Then

        Add 1 to counter containers that cannot be (re)located

    End If
End Sub
```

Figure 23: Pseudocode subs (Re)LocateContainers()

APPENDIX B: VISUAL REPRESENTATION YARD BAY

							Containers in row
1st row						2	1
						2	1
						2	1
						2	1
				2		2	2
				2		2	2
Stack ht	0	0	0	2	0	6	
2nd row					2		1
					2		1
			9		2		2
			9		2	1	3
			9		2	1	3
	18	10	9	11	2	1	6
Stack ht	1	1	4	1	6	3	
3rd row				5			1
				5			1
	2			5			2
	1			5		4	3
	3	0		5		4	4
	3	0	18	5		4	5
Stack ht	4	2	1	6	0	3	
4th row				4			1
				4			1
				4			1
				4			1
				4			1
			1	4		3	3
Stack ht	0	0	1	6	0	1	
5th row	1		2		1	1	4
	1		2		1	1	4
	1		2		1	1	4
	1		2		1	2	4
	1		6		1	2	4
	1		6		1	1	4
Stack ht	6	0	6	0	6	6	

Figure 24: Visual representation yard bay

COFANO

APPENDIX C: COMPLETE RESULTS

Stacking method: 156 days run

Random	Run	Arrivals	Retrievals	Relocations	Relocation	Avg. capacity	max capacity	No location	No relocation	Run-time(s)
Big (3000)	*1st*	4466	4257	4760	35.30%	45.29%	70.60%	738	157	79.97
	2nd	4465	4261	4599	34.51%	44.60%	67.80%	739	148	121.93
	3rd	4489	4266	4466	33.78%	44.68%	66.70%	715	162	97.56
	4th	4474	4284	4587	34.37%	45.11%	64.40%	730	129	106.29
	5th	4392	4153	4575	34.87%	44.58%	62.80%	812	181	152.8
	Avg	*4457.2*	*4244.2*	*4597.4*	*34.57%*	*44.85%*	*66.46%*	*746.8*	*155.4*	*111.71*
Medium (2000)	*1st*	3325	3238	3445	34.42%	35.40%	63.90%	136	40	102.85
	2nd	3322	3208	3312	33.65%	35.07%	66.70%	139	67	109
	3rd	3345	3238	3370	33.86%	35.22%	63.90%	116	60	71.69
	4th	3294	3203	3341	33.96%	34.97%	58.90%	167	44	51.53
	5th	3324	3223	3241	33.11%	35.20%	64.40%	137	54	54.27
	Avg	*3322.0*	*3222.0*	*3341.8*	*33.80%*	*35.17%*	*63.56%*	*139.0*	*53.0*	*77.87*
Small (1000)	*1st*	1709	1683	1408	29.33%	18.65%	41.70%	0	0	24.89
	2nd	1709	1683	1257	27.04%	18.65%	41.70%	0	0	25.38
	3rd	1709	1683	1306	27.80%	18.65%	41.70%	0	0	26.34
	4th	1709	1683	1318	27.98%	18.65%	41.70%	0	0	27.06
	5th	1709	1683	1359	28.60%	18.65%	41.70%	0	0	28.35
	Avg	*1709.0*	*1683.0*	*1329.6*	*28.15%*	*18.65%*	*41.70%*	*0*	*0*	*26.40*
Average		*3162.7*	*3049.7*	*3089.6*	*32.17%*	*32.89%*	*57.24%*	*295.3*	*69.5*	*72.0*

Current	Arrivals	Retrievals	Relocations	Relocation	avg capacity	max capacity	No location	No relocation	Run-time(s)
Big (3000)	4824	4677	4933	34.18%	51.01%	79.00%	380	76	292.42
Medium (2000)	3441	3387	3465	33.66%	37.62%	73.00%	20	7	206.66
Small (1000)	1709	1683	1303	27.75%	18.65%	42.00%	0	0	97.43
Average	*3324.67*	*3249*	*3233.67*	*32%*	*36%*	*65%*	*133.33*	*27.67*	*198.84*

New	Arrivals	Retrievals	Relocations	Relocation	avg capacity	max capacity	No location	No relocation	Run-time(s)
Big (3000)	4865	4744	2292	19.26%	51.04%	86.00%	339	56	78.45
Medium (2000)	3427	3372	879	11.45%	37.30%	73.00%	34	8	94.58
Small (1000)	1709	1683	120	3.42%	18.65%	42.00%	0	0	28.21
Average	*3333.7*	*3266.33*	*1097*	*11.37%*	*35.66%*	*67.00%*	*124.33*	*21.33*	*67.08*

Yard Crane (Current)	Arrivals	Retrievals	Relocations	Relocation	avg capacity	max capacity	No location	No relocation	Run-time(s)
Big (3000)	5028	4956	3581	26.40%	54.36%	94.00%	176	0	145.68
Medium (2000)	3461	3414	2476	26.48%	37.98%	79.00%	0	0	105.84
Small (1000)	1709	1683	1223	26.50%	18.65%	42.00%	0	0	55.71
Average	*3399.3*	*3351*	*2426.67*	*26.46%*	*37.00%*	*71.67%*	*58.67*	*0*	*102.41*

Yard Crane (New)	Arrivals	Retrievals	Relocations	Relocation	avg capacity	max capacity	No location	No relocation	Run-time(s)
Big (3000)	5187	5075	63	0.61%	56.79%	100.00%	40	0	90.05
Medium (2000)	3461	3414	0	0.00%	37.98%	79.00%	0	0	64.54
Small (1000)	1709	1683	0	0.00%	18.65%	42.00%	0	0	35.66
Average	*3452.3*	*3390.67*	*21*	*0.20%*	*37.81%*	*73.67%*	*13.33*	*0*	*63.42*

Table 14: Complete results