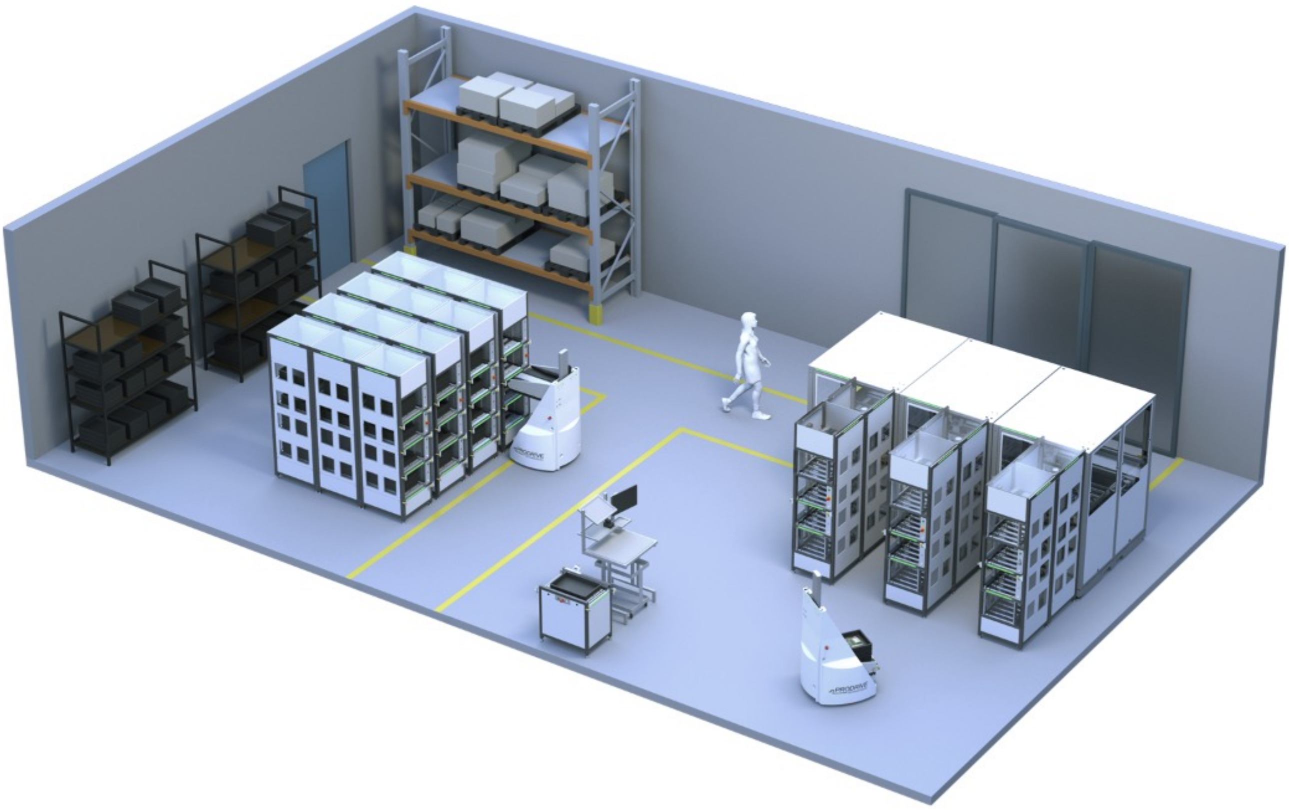
The design of flexible autonomous bin handling systems

T.A.N. Maaskant

Design strategies
for bin handling systems
in production environments

ISBN 000-00-0000-000-0



TU Delft

PRODRIVE TECHNOLOGIES

Master Thesis

TU Delft, 3mE - MME

The design of flexible, autonomous bin handling systems in a production environment

By

Tim Maaskant

in partial fulfilment of the requirements for the degree of
Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology of Faculty Mechanical,
Maritime and Materials Engineering of
Delft University of Technology
To be defended publicly on *Date and time*

Student number:	4473221	
MSc track:	Multi-Machine Engineering	
Report number:	2022.MME.8805	
Project duration:	September 12, 2022 – date	
Thesis committee:	F. Schulte,	TU Delft, committee chair
	Dr. W. W. A. Beelaerts van Blokland,	TU Delft, supervisor
	Ir. K. van den Elsen,	Prodrive Technologies, supervisor
	Ir. L. de Wit,	Prodrive Technologies, supervisor

Preface

This thesis is the outcome of my graduation assignment at Prodrive Technologies. Here I have had the opportunity to study a state of the art logistics automation system. The bin handling system at Prodrive was developed in house, which means all knowledge of the system was present in the company. During the 8 months of this research, I have constantly found new angles on how to tackle problems and view larger multi agent systems. As I am highly interested in the field of logistics automation, I am grateful for the chance to have worked with such a high tech system and broaden my understanding of such systems.

I would like to thank my supervisor Wouter Beelaerts van Blokland, from TU Delft, for his support and guidance. Wouter has given me insights in new angles which would ultimately shape my research further. I would also like to thank my supervisors from Prodrive Technologies, Koen van den Elsen and Leon de Wit. They were exceptionally helpful, they brought creativity and critical questions which made my research more complete. Lastly, i would like to thank the people around me, my friends, parents and especially my sisters, who carried me through tough times when needed.

T.A.N. Maaskant
Eindhoven, April 2023

Abstract

Autonomous, flexible bin handling systems are state of the art. Designing such systems therefore is often done with few examples or knowledge of performance alteration design strategies. In the system development cycle, most of the costs are committed in the conceptual design phase. Having knowledge to base design choices on is therefore crucial. This thesis aims to give insight in different design choices of bin handling systems.

During this thesis, a case study is done at Prodrive Technologies. Prodrive Technologies is heavily invested in building the factory of the future, which has been called: the light out factory. As the name suggests, it is a factory in which the light can be turned off since there are no more humans required. In order to achieve this goal, an autonomous bin handling system had been developed. This system handles all movement of goods within the production environment. It consist of easily employable units, so no conveyor belts designed for one specific line or product. This means the system can be employed in different environments without the need for modification of its units.

In this thesis, performance is defined in terms of system costs, system footprint and demand scalability. To evaluate the performance, a quantitative model is constructed and solved using Gurobi Optimization software. The model determines the least possible amount of storage space required in the system. Hereafter, it selects the units which are most cost, or floorspace effective to use and assigns a location to those units. This quantitative model also generates a design for the layout of the system, fitted in the production environment.

A method for system footprint determination is proposed in this research. Placement of units in the environment introduces areas where no other activities can take place. The method proposed not only counts the footprint of individual units, but also the areas restricted from other use due to the placement of the units.

Designs are proposed by identifying the different functions of bin handling systems. Next these functions are combined in different units in order to see how performance changes. This is done relative to the case study system. Since bin handling systems have a storage stage, storage strategies of warehouses are reviewed in this research. Their effectiveness on the storage stages are evaluated.

Nine designs have been evaluated in this thesis. These designs are tested for total cost, bill of material cost and floorspace optimal solutions. All optimal solutions are then generated again for higher production demands in order to gain insight in the validity of designs for higher production speeds. It is found that the following design choices have the following impact:

- Shared storage strategies have a negligible effect on both costs and floorspace
- The storage stage only effects a small amount of the costs and floorspace
- Transporting bins instead of stacks is a viable strategy, but will increase system footprint
- Combining the transportation and positioning function promises the largest decrease in total costs, while not increasing footprint
- Small local buffers in addition to central storage reduce costs and footprint
- Larger local buffers will increase total systems costs.

Contents

List of Figures

List of Tables

List of abreviations

Abreviation	Explaination
AGV	Automated guided vehicle
API	Application programming interface
BOM	Bill of material
BPS	Bin processing solution
DP	Dynamic programmin
DSCAB	(de)Stacker cabinet
DSM	(de)Stacker module
FIFO	First in first out
FTE	Fulltime equivalent
GAP	Generalized assignmet problem
HBT	Horizontal bin transport
KPI	Key performance indicator
LES	Logistics execution system
LIFO	Last in first out
LP	Linear programming
MBC	Multi bin cabinet
OR	Operations research
PEL	Payload exchange location
SGPS	Second generation particle sensor
SPEL	Single payload exchange location
TSP	Traveling salesman problem
USL	Unique storage location
V&V	Verification and validation
VBT	Vertical bin transport
VRP	Vehicle routing problem

1 Introduction

This chapter provides a general introduction to this thesis on the design strategies of bin handling systems. First some background information on industrial automation and logistics automation is given. Next in section 1.2 the concept of a bin handling system is explained. In section 1.3 the research motivation is discussed. In section 1.4 the research questions are discussed. The research scope is defined in section 1.5 and lastly in section 1.6 the outline of the report is presented.

1.1. Background

Automation of manufacturing processes started with the steam engine and the famous conveyor belt systems. Since then automation of manufacturing process started to focus more on the manufacturing steps of the process. Robots carrying out tasks autonomously, reducing the need for human operators more and more. In the current day and age robotisation in production lines is nearing its limit. The focus now lies on optimization and communication properties of robots. When looking at the whole production chain however, there are still gaps where robots are not present. The most obvious one is the intralogistics field.

Intralogistics is a field of logistics which includes all goods and material flows happening on the company premises. Company premises are a highly dynamic environment, meaning there are variables that are hard to control and are constantly changing. Think of employees walking around, products moving from A to B or obstacles in pathways. Navigating such an environment is mostly done by humans since they are capable of assessing different situations and react accordingly.

The next step in fully automating manufacturing processes is the one of intralogistics. The main task here is transporting sub assemblies to an assembly line and retrieving and storing finished products. Now done by human operators, large pallets of sub assemblies are taken from the warehouse and transported to the assembly line. There pallets are unpacked and the individual raw materials or sub assemblies are loaded into the respective production cell. This leaves room for error from the operator by supplying the cell too late, or with the wrong component. Automating this process reduces the chance or errors and increases the autonomy of the process. It is not by accident that intralogistics are not yet widely automated. Automating the movement of goods is challenging. As stated above, company premises are highly dynamic environments and are therefore hard to control. This means robots that are capable of moving goods must also be able to navigate this dynamic environment. Advancements in sensing, communication and control technologies enable robots to carry out such advanced tasks with a high level of autonomy. Such robots are formally known as automated guided vehicles (AGV).

While one could consider the autonomous guided vehicle as the base of logistical automation, it still needs other machines to interact with. An autonomous logistics system must be able to perform more tasks than just driving a product from A to B. Tasks such as feeding, picking, (de)stacking and more should also be considered when eliminating all operators from the process. Communication between different robots is key here. By communicating with each other or a central command center, the robots can create a planning themself resulting in a truly autonomous system.

When operating at the highest level of autonomy, no humans should be required to run a production line. Realising such an environment is the long term goal of Prodrive Technologies in Eindhoven. Prodrive technologies is a high-tech company in the Netherlands that develops and produces mechatronic and electronic systems and products. To improve capabilities of their production processes, a so called 'light-out factory' is pursued. This lights-out factory has such a degree of autonomy it can be left alone for a shift, hence the name. This research is conducted in cooperation with Prodrive Technologies in order to improve the implementation and further development of their autonomous bin processing system.

1.2. System definition

To enable transport of various products, one might opt for a generalized transport container. This helps in the development of product handling systems since only one type op container needs to be handled. Such generalized transport containers are called bins. Any number and types of machines and robots can be used to transport these bins from A to B. Most intralogistic movements are between warehouses and production lines. These movements can be divided into two different processes: heavy lifting and single supply. Heavy lifting is the movement of large amount of products, pallets or stacks of smaller bins, over longer distances. Often this is to supply the whole production environment with enough goods. Single supply is the servicing of a single production cell. This thesis focuses on autonomous bin handling systems supplying the production cells. Since flow of goods is not constant, such systems often also include storage buffers.

1.2.1. Use case at Prodrive Technologies

For this research a case study is done at Prodrive Technologies. At Prodrive Technologies an autonomous bin handling system has been developed and is operating in a real world production environment. This system consists of four distinctly different units, which are illustrated in figure 1.1. First is the single payload exchange platform (1). This unit handles the introduction and extraction of products in and out of the system environment. Next is the multi bin cabinet (2). This is a multi layer storage unit capable of being expanded backwards. Here the majority of storage takes place. The (de)stacker cabinets (3) are used the stack and destack bins filled with products and are commonly placed next to production cell to supply them with single bins. Lastly there are the automated guided vehicles (4). These are the transporters of products within the environment. They are capable of docking to other units and exchanging products. This system will be used as a case study to evaluate the costs of implementing this system as is, and how these costs can be decreased.

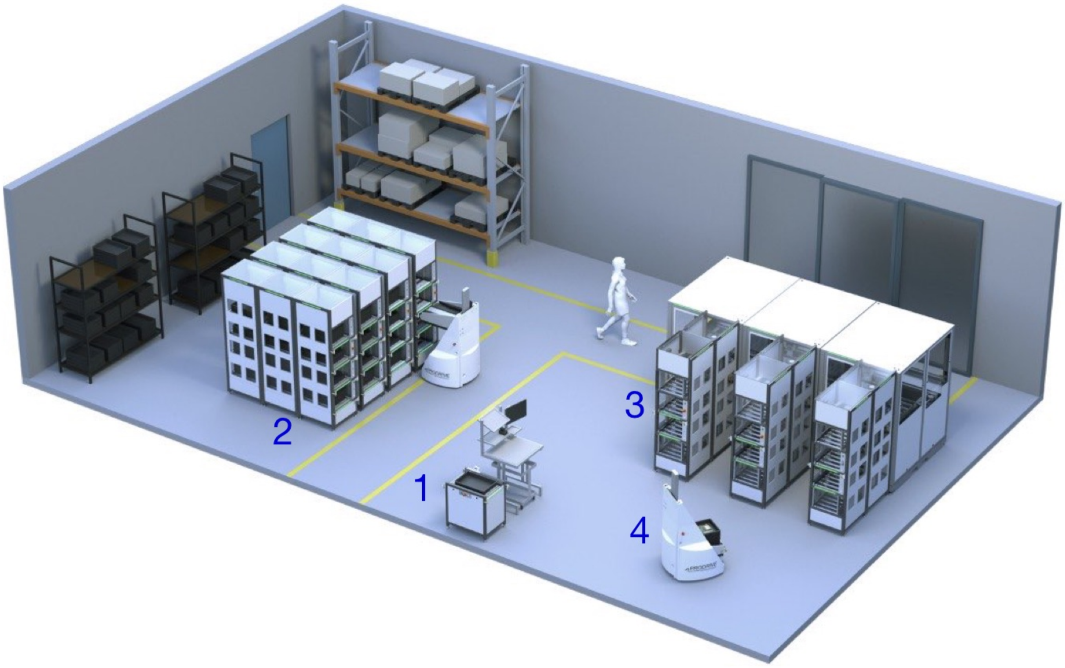


Figure 1.1: The bin handling system of Prodrive Technologies

1. Single payload exchange platform

2. Multi bin cabinet

3. (de)Stacker cabinet

4. Automated guided vehicles

1.3. Research motivation

Fully autonomous bin handling systems are only bound by imagination. One could make one robot capable of carrying out all tasks required by the system, or one robot for each individual task. Since such systems are state of the art there are not many examples of existing systems. This is undesirable since 75 % of the costs are committed during the design phase as can be seen in figure 1.2 by Baral, 2021. While the costs may come later, choices in the design phase will determine if, and in what degree they come. Being well informed on performance expectations, and having a way to quantify them, early in this process can save money, time and effort later on. Not only system costs but also floorspace and performance of such a system should be considered. Developing a system not capable of supporting growing demand means more costs later on due to adjustment costs of excessive floorspace requirements. This thesis aims to **provide quantitative insight in the effects of different design solutions for autonomous bin handling systems, so better informed decisions can be made in the design phase**.

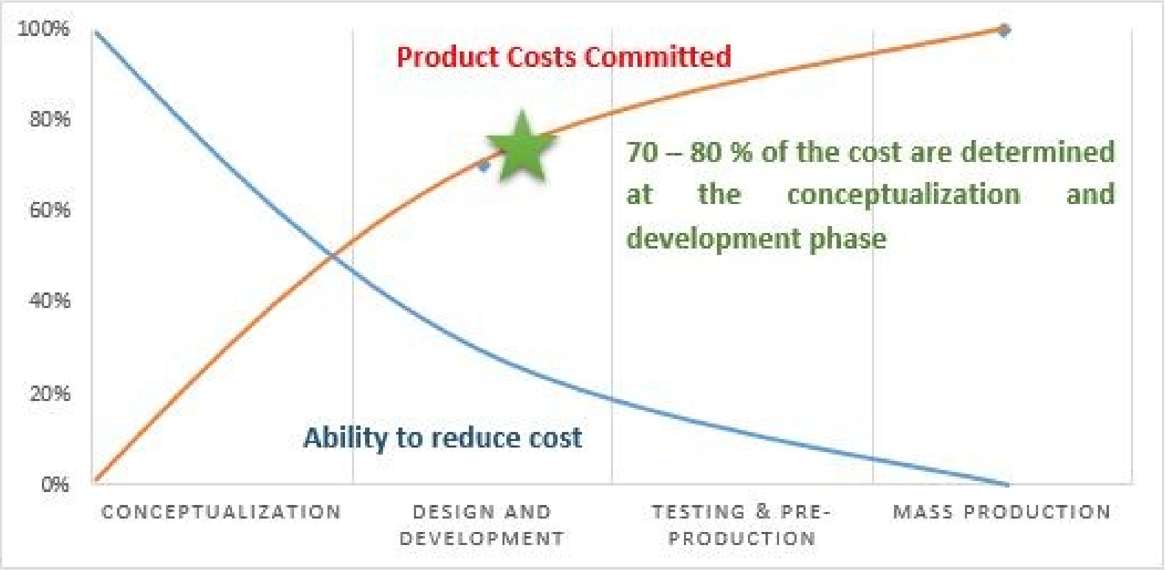


Figure 1.2: Product costs in the product development cycle, Baral, 2021

1.4. Research questions

This thesis aims to evaluate the performance of various autonomous bin handling systems. More specifically, optimizing the usage of autonomous buffer systems. The research question this thesis aims to answer is:

What are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?

To be able to answer this research question, the following sub questions are formulated:

1. What is the definition of buffering and how does this relate to warehousing?
2. What are the design strategies present in existing bin handling systems?
3. How can a cost optimal solution be found using quantitative modeling?
4. How is a bin handling system defined and how do the costs relate to the system?
5. How does a quantitative cost optimal solution of a bin handling system relate to an analytical solution?
6. What is the impact of different design strategy?

1.5. Research scope

This research focuses on both the cost and floorspace effects of design choices bin handling systems, as well as how the design choices cope with an increasing demand. Since bin handling systems can vary in shape and size, the variables considered by this research should be clearly defined. The bin handling system consist of three main aspects: the hardware, the operational software and the placement in the production environment (layout). There are more components to consider but they are defined as external factors. External factors are considered to be inputs and unchangeable. These factors include the production line layout and the logistical execution system (LES).This is depicted in figure 1.3.

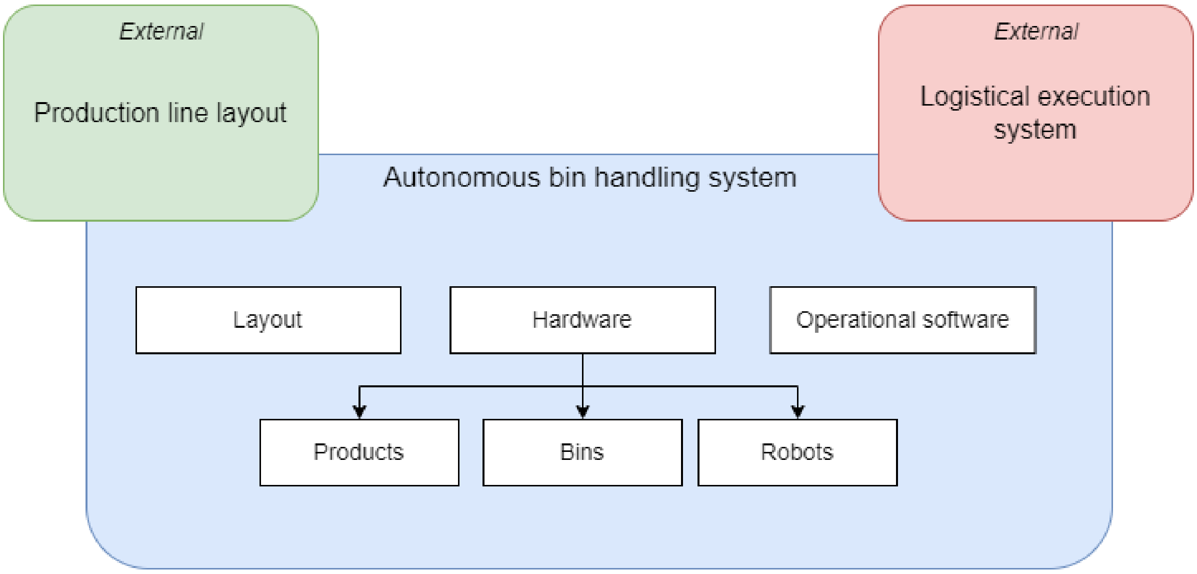


Figure 1.3: Aspects of the autonomous buffer system

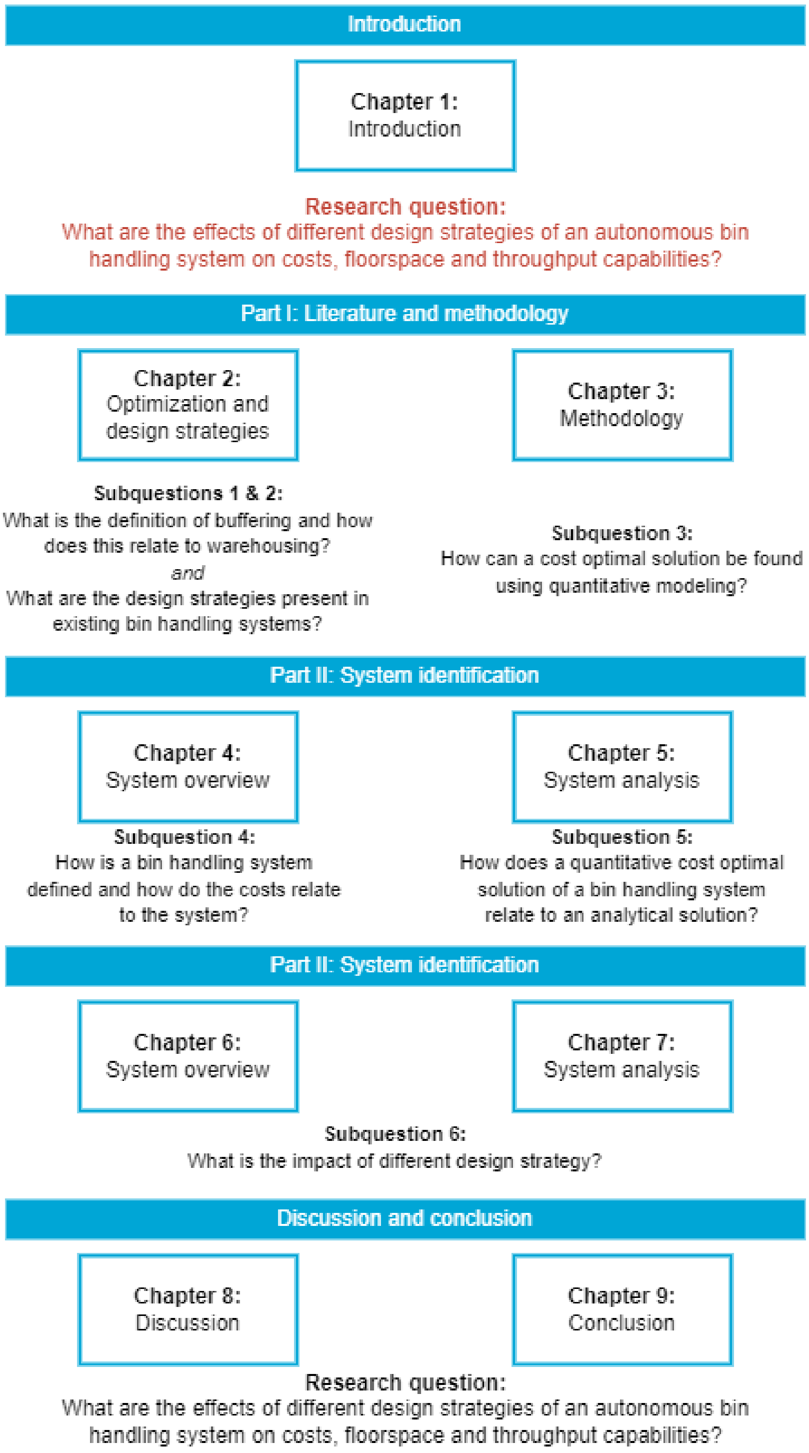
Every autonomous system is operated by operational software. This software dictates how robots move and interact with each other. This does not include operational planning since this is handled by LES. Improvements in operational software could improve overall performance. This however is more software related and not in the field of this thesis. Therefore, operational software is excluded from this scope. Since operational planning software is excluded, route optimization is also excluded from the scope.

Placement, or layout, of the bin handling system units is relevant in a few ways. First of which is the route planning of the AGVs. More efficient route planning can decrease the travel time of AGVs, in turn leading to fewer AGVs or a higher potential throughput. Due to relatively long (un)docking operations and high travel speeds of the AGV, a decrease of travel distance is not significant in total job time. To put this into perspective, a 20 % decrease in travel distance due to layout optimization will result in a 1 % total total job time reduction. This is based on the case study system described in section 1.2.1. The second way placement of units is relevant is the floorspace the systems requires. Since floorspace is one of the key performance indicators of this thesis, placement with respect to floorspace optimization is within the scope.

The last part of an autonomous bin handling system is the hardware. This also includes the products and bin shape the systems should handle. Since the scope would be too wide if these factors are considered, these factors are seen as a constant. Products cannot be altered to fit bins better, and bins cannot be altered to improve capabilities of the robots.

The focus of this thesis is to quantify the effects of different design choices of robots present in a bin handling system. A bin handling system is required to be capable of different tasks. The number of different robots required for those tasks is up to the designer of the system. Less robots might save costs but could hurt the throughput of the production line. The main scope of this thesis is the influence of hardware, both the type and quantity of the robots, on the costs, floorspace and throughput of the system.

1.6. Report outline



Introduction

Chapter 1: Introduction

Research question:
What are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?

Part I: Literature and methodology

Chapter 2: Optimization and design strategies

Chapter 3: Methodology

Subquestions 1 & 2:
What is the definition of buffering and how does this relate to warehousing?
and
What are the design strategies present in existing bin handling systems?

Subquestion 3:
How can a cost optimal solution be found using quantitative modeling?

Part II: System identification

Chapter 4: System overview

Chapter 5: System analysis

Subquestion 4:
How is a bin handling system defined and how do the costs relate to the system?

Subquestion 5:
How does a quantitative cost optimal solution of a bin handling system relate to an analytical solution?

Part II: System identification

Chapter 6: System overview

Chapter 7: System analysis

Subquestion 6:
What is the impact of different design strategy?

Discussion and conclusion

Chapter 8: Discussion

Chapter 9: Conclusion

Research question:
What are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?

I

Literature and methodology

2

Optimization and design strategies

This thesis aims to provide a method to generate an optimal solution for bin handling system implementations in a production environment. In this section methods for optimizing storage hardware usage are researched. Since bin handling systems are state of the art, a broad search has been done. Methods for warehouse management are also included since they optimize storage capacity. This can be seen in section 2.1. Next logistical optimization methods are researched. Focus here lies on the amount of ”hubs” since storage equipment can be seen as hubs. this can be read in section 2.2. The difference between buffering and warehousing should be clear. This is explained in section 2.3. Lastly a search for literature on existing bin handling systems is done.

2.1. Warehouse or buffer optimization methods

While bin handling systems are more than just storage, storage components are required in order to ensure production is not halted. These storage components can be seen buffers or small warehouses. All problems related to warehousing and buffering should therefore be explored. Order picking is the most costly operation in a warehouse Marchet et al., 2015. Traditionally it includes employees navigating the warehouse and picking items according to customer demands. Since this is costly, many studies have been done into optimizing this process. Van Gils et al., 2018 identifies two main categories of problems in order picking: tactical and operational. This is depicted in figure 2.1. Designing an optimal order picking system has to weigh each of the depicted aspects. Optimizations can be done to fit different requirements such as warehouse space, travel distance, storage space and picking speed. Depending on the optimization goal, order picking optimization can be used for more situations than picking multiple different items and delivering them to customers.

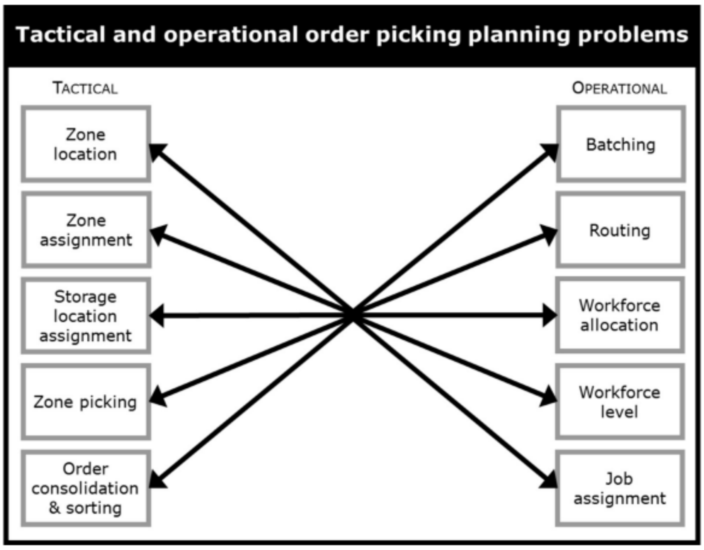


Figure 2.1: Aspects of order picking, Van Gils et al., 2018

2.1.1. Storage assignment problem

The storage assignment problem is part of order picking optimization. R. De Koster et al., 2007 describes storage assignment as decisions on how and where a set of items is stored in order to optimize the logistic system. Even though there are numerous methods to assign products to a storage location, there are five most frequently used methods: random storage, closest open location storage, dedicated storage, full turnover storage and class based storage. Random storage methods randomly select a storage location for an item. This results in high travel times. Due to the constantly changing locations of items these methods only work in computer controlled environments as stated by Quintanilla et al., 2015. Closest open location methods find the closest empty storage location and use it for the storage of the held item. This results in a concentration of products near the handling area, meaning lower traveling time. Closest to empty and random methods can also be used in combination with each other. This is done through using a non-uniform distribution when selecting a random location as stated by Park, 1987.

Dedicated storage methods designate one specific storage location to one specific item. For larger warehouses, where not all items are always on stock, this means not all storage space is needed at one time thus requiring a larger warehouse. However, by having a designated location for each item the planability for the logistic system increases. This is especially true when assuming no item is out of stock. Then the larger storage locations can be reserved for items moved in larger quantities, thus reducing the overall number of storage locations needed. M. De Koster and Neuteboom, 2001 states designated storage locations reduce work and mistakes in human operated stores since each item is always in the same place, thus reducing search times. While computer directed systems could have the ability to directly communicate new item locations, this is not always a given. For less advanced computer systems without the capabilities of real-time communication dedicated storage also makes them less likely to make a mistake. Decisions on where the dedicated storage for an item will be assigned can be based on multiple different aspects such as part number, barcode number or duration of stay

Full turnover methods focus on the placement of items according to demand. Placing items with a higher turnover at more easily accessible storage locations. This method is especially usefull when optimizing average item retrieval speed, Yang et al., 2017. Full turn over is an expansion on dedicated storage. Yu and De Koster, 2013 states that full turnover suffers the same disadvantage as dedicated storage of needing storage spaces able to accommodate the full inventory of a single item. To evaluate the demand of an item the cube-per-order index (COI) is commonly used. Proposed by Heskett, 1963, the COI is defined as the ratio of maximum allotted space to the number of storage/retrieval operations per unit time, Bahrami et al., 2019.

Class based storage divides all stock items into classes and stores them in the same area. Storage locations are then randomly assigned within an area. As mentioned by Bahrami et al., 2019, classes can be defined as seen fit. High turnover classes can be located near handling areas, while low turnover classes can be located further away. This results in lower traveling times. The random storage assignment in an area reduces the need for large storage areas and makes warehouses more flexible when demand changes.

2.1.2. Comparison of discussed methods

In this section we looked for an answer to our subquestion: how can the storage locations be assigned? In table 2.1 the discussed methods are summarized and in figure 2.2 a graphical representation is shown on how the different methods are related. We see for different applications different methods are more usefull. A big factor in selecting the right methods in certain situations, are the capabilities of the system. Random storage methods are only applicable if the system can register and keep track of where items are placed. Even though human operated systems are capable of registering item placement changes, it often leads to longer order picking times and confusion. Random item placement increases the flexibility of the warehouse. Fluctuations in stock amount or item types can be accommodated. A big flaw in dedicated storage types is the amount of storage space needed. When assuring there is always a large enough storage location available for an item, some locations will be empty when the stock for that item is low. This is exceptionally bad in warehouses where items change often or stock is not always guaranteed. However, for warehouses where stock is always required to be at maximum capacity dedicated storage uses the available storage space optimal.

	Flexibility	Chaotic	Use of storage space	Planability	Traveling time
Random location	High	Very	Optimal	Low	High
Closest open location	High	Somewhat	Optimal	Low	Low
Dedicated location	Low	No	Low [1]	High	High
Full-turnover	Low	No	Low [1]	High	Low
Class based storage	Moderate	Somewhat	Moderate	Moderate	Low

Table 2.1: Comparison of discussed storage assignment methods

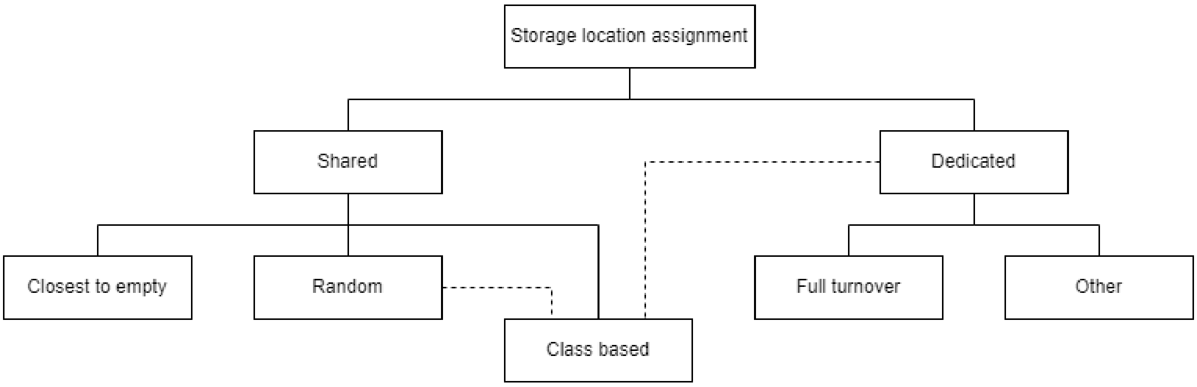


Figure 2.2: Overview of the different storage location assignment methods

2.2. Operations research

Operations research (OR) is a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control, Morse et al., 2003. Rather than intuitive or experience based decisions, OR has a scientific base. This means methods are comprised for analysing different situations in different fields. These fields include management, government, military, logistics and more. For our applications we are interested in the logistics aspect of operations research.

[1] Optimal if assuming always full stock for all items

Within the logistics OR field there are multiple problems defined and methods related to those problems. Each method has its own specific use. In the subsection below the methods that might be of interest to our research are elaborated.

2.2.1. Transportation and shortest path problem

The transportation problem focuses on the travel time minimization. As stated by Nikolić, 2007: there are two types of problems regarding the transportation time: (i) minimization of the total transportation time (linear function, as aggregate the products of transportation time and quantity), called minimization of 1st transportation time, and (ii) minimization of the transportation time of the longest active transporting route (nonlinear function), called minimization of 2nd transportation time or problem of Barasov, 1961. Transport problems are mainly found in supply chain scheduling. In real life transportation problem there are of a lot of uncertainties. Here a stochastic, Williams, 1963 or fuzzy, Singh and Yadav, 2016 approach is therefore often a good choice.

The shortest path problem is related to the transportation problem with a small difference in optimization goal. The shortest path problem is looking to minimizing the total traveled distance. The definition is stated by Magzhan and Jani, 2013 as: a problem of finding the shortest path or route from a starting point to a final destination. This can also include mandatory stops. The shortest path problem is often displayed as a graph as shown in figure 2.3. For less complex graphs the solution can be calculated intuitively. For more complex situations methods have been created such as the Dijkstra's algorithm and Floyd-Warshall algorithm.

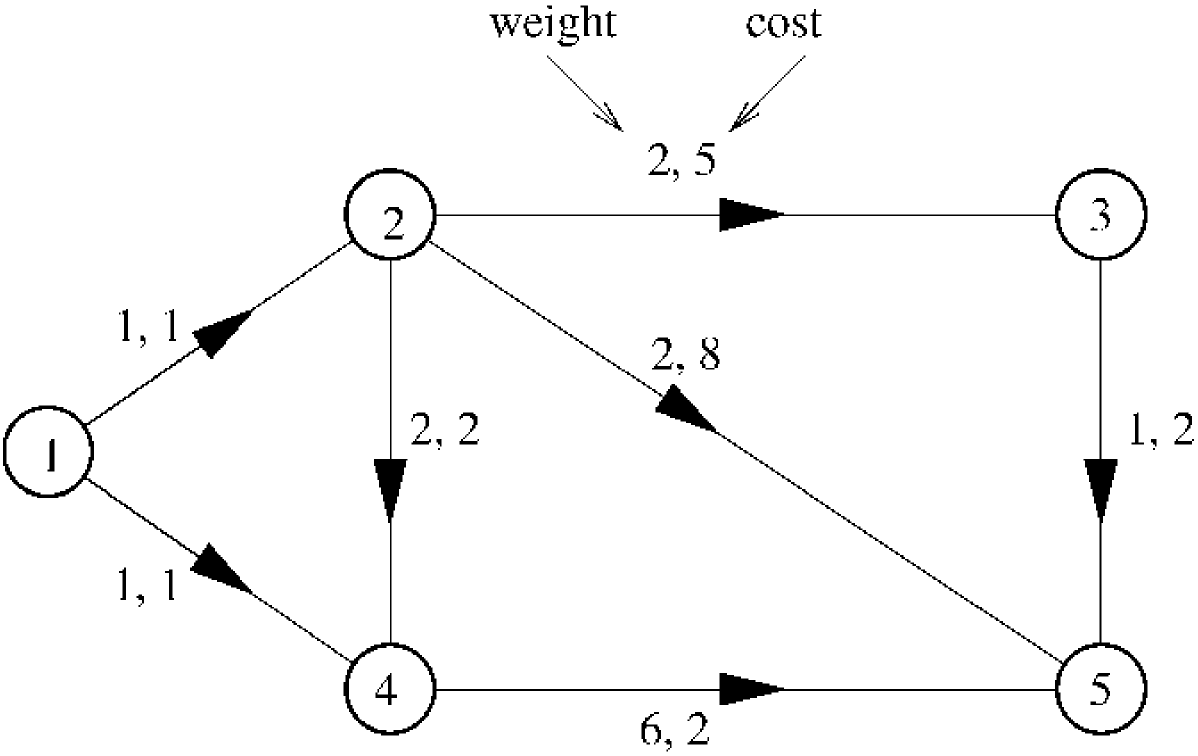
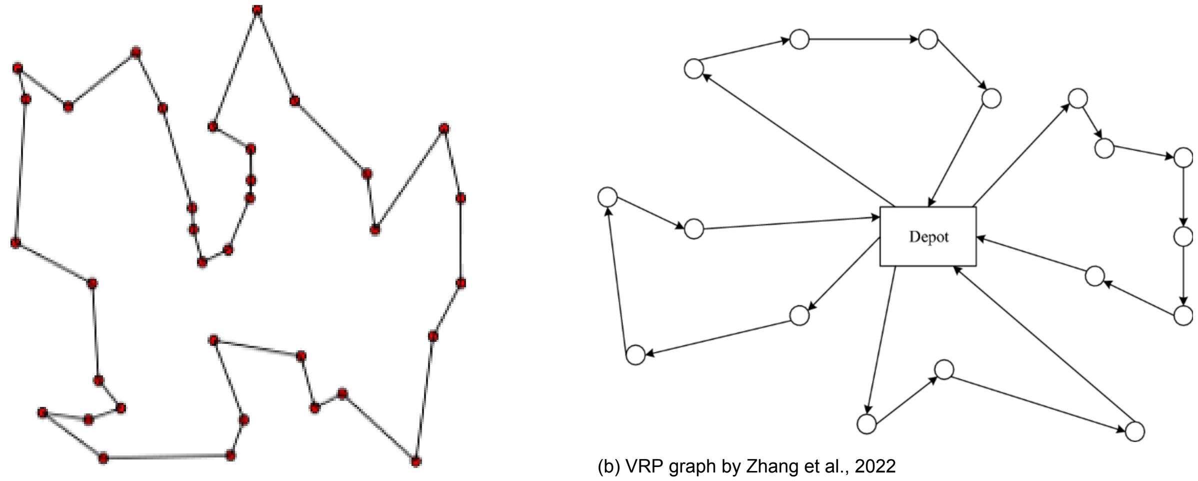


Figure 2.3: Example of a shortest path problem graph by Dumitrescu and Boland, 2001

2.2.2. Traveling salesman and vehicle routing problem

The traveling salesman problem (TSP) originates from the question: : if a traveling salesman wishes to visit exactly once each of a list of X cities and then return to the home city, what is the least costly route the traveling salesman can take,Hoffman et al., 2013. In contrary to the shortest path problem, TSP want to find the overall shortest route without visiting a node twice. In TSP the order of nodes visited does not matter. A good example of a TSP is the delivery of parcels. A delivery driver does not want to visit the same street twice. His route should be selected in such a way this does not happen. This is an optimization of the travel distance, not of the amount of equipment (vehicles) needed to meet the demand, Jünger et al., 1995.

Similar to the TSP are vehicle routing problems (VRP). The biggest difference is in TSP only one vehicle is considered. In VRPs the goal is to distribute goods between depots and and final users as efficiently as possible, Toth and Vigo, 2002. Here the term 'efficient' means at the lowest cost. Since VRPs consider the whole system, the amount of equipment needed can also be varied. This means there is not only a minimization of travel time or distance. For example: it might be cheaper to use less vehicles with longer routes instead. VRPs can be used to determine: amount of depots, optimal depot locations, amount of vehicles needed and routing costs, Pillac et al., 2013. In figure 2.4 the difference between TSP and VRP is depicted.



(b) VRP graph by Zhang et al., 2022

(a) TSP graph by Glen, 2022

Figure 2.4: Graphical representation of different methods

2.2.3. Generalized assignment problem

The generalised assignment problem (GAP) is the problem of finding the minimum cost assignment of n jobs to m agents such that each job is assigned to exactly one agent, subject to an agents capacity, Chu and Beasley, 1997. GAP is often used in production planning. Using a certain high end machine might be costly but its throughput might also be high. Planning to produce a small batch on this machine will then increase the costs since now the machine is not available anymore for larger batches of products. In GAP the goal is to minimize the costs through optimal use of equipment for the jobs at hand.

2.2.4. Queueing theory

The last field of logistic operations research we discuss is queueing theory. In queueing theory the best possible ways to handle customers and assign them to queues are discussed, Newell, 2013. Queueing often is a stochastic process. This is due to multiple reasons such as different arrival and service rates. The example of customers is common but queueing theory can also be used in warehouses or computer science in order to optimize flow. Optimization is done into waiting times and partly into resource minimization.

2.2.5. Summary of discussed methods

Since logistics is a process with multiple steps, logistic operations research has multiple focus points. In table 2.2 the different focuses of the different methods are summarized. There are three distinct aspects: travel/transportation, equipment and waiting time. Depending on the specific situation, different methods could be used or combined.

Method	Focus
Transportation	Minimization of total transportation time
Shortest path	Minimization of total travel distance
Traveling salesman	Minimization of total travel distance
Vehicle routing	Minimization of total transportation costs including equipment
Generalized assignment	Minimization of equipment
Queueing theory	Minimization of waiting time and equipment

Table 2.2: Focuses of different operations research topics

2.3. Buffering vs warehousing

The term 'buffer' is defined as a safety margin to not disrupt a process when components fail and ensure continuity. In supply chain systems a buffer is a small, local stock of items to accommodate fluctuations in supply rate. From management strategies such as lean manufacturing, Gupta and Jain, 2013, we learn that buffer systems increase the costs of any system. Meaning in an ideal world buffer systems are as small as possible. This raises another question: what is as small as possible? A to small buffer may not be able to accommodate fluctuations and will hold up the stream. If the buffer size is too large, excessive space is allocated for hardware and inventory, it may result in unnecessary costs, although the process reliability may improve. In our case the buffer system is not only used to ensure continuity but also as a forward warehouse. Meaning production should be able to continue for X amount of time without the need for an operator to resupply. Right of the bat it is clear the X amount of time will be a cost driver. Since keeping stock for a full shift demands much more items being in the buffer than for one hour.

The type of buffer in a bin handling system is subject to different requirements then a warehouse, Umirzoqov, 2020. In production environments, all demand will always be met. Starting production with one part of the product not in stock will hold up production immediately. The number and quantity of products is known in advance, and will be constant for each shift. This makes it possible to perfectly tune the size of the buffer capacities of the bin handling system. Problems emerge when one production line can produce multiple different products, Mahadevan and Narendran, 1993.

2.4. Design parameters of bin handling systems

The design parameters of a bin handling system are to be defined by their application. However, some design choices are more cost effective. The different design parameters of existing systems are to be identified in order to answer the subquestion: *What are the design strategies present in existing bin handling systems*

Mahadevan and Narendran, 1993 divides buffers in central and local buffers. Central buffers being storage locations available to all products and units. Local buffers are buffers integrated in a unit, specifically for one type of product. This is schematically illustrated in figure 2.5. Local buffers can be optimized for specific products, but when flexibility or a broader employability of the system is required this kind of buffer may require refitting.

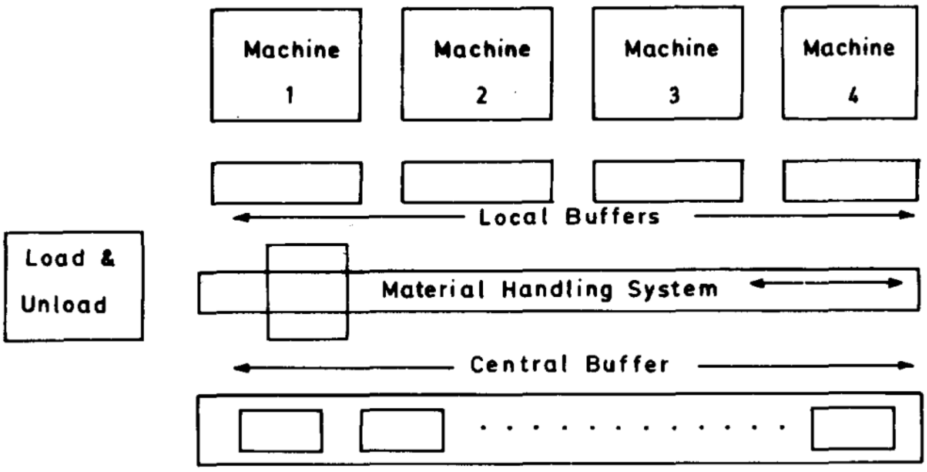


Figure 5. Layout of the system considered for the current study

Figure 2.5: Schematic view of buffer types and their locations, by Mahadevan and Narendran, 1993

As noted by Furmans et al., 2010, bin handling systems are often inflexible due to components such as conveyor belts. Newer systems therefore operate in more dynamic environments with fast changing production line configurations. While facing new challenges, the main operating principles remain the same from traditional bin and material handling systems. Kay, 2012 formulates five different major equipment types.

1. Transport equipment. Equipment used for transportation of goods within the system from one location to another.
2. Positioning equipment. Equipment handling products at a single location in order to position products for further steps, often located in front of production cells.
3. Unit Load Formation Equipment. Equipment used to secure materials and preserve their integrity during transport and storage. In bin handling systems these are often the moulds in the bins, keeping the product in place in the bin.
4. Storage equipment. Equipment capable of storage for short, or sometimes longer, periods of time. Often used as buffers to cope with supply fluctuations or the storage of parts needed for production near the production environment.
5. Identification and Control Equipment. Equipment utilized for gathering and transmitting information, crucial for coordinating the movement of materials within a facility.

While Kay, 2012 focuses on traditional material handling system equipment, newer autonomous systems aim to consider functions and how they can be combined in a broader range of applications. These systems however are novel and are few and far between. Some papers such as Ye et al., 2018 talk about the use of AGVs for product and bin handling, but not in a production environments setting. However, Ye et al., 2018 does explain the combining of different function in one unit. Here, the transport and positioning function are combined.

2.5. Chapter summary

In section 2.1 the methods of warehouse and buffer optimization are researched. In the literature found, there is hardly any mention of buffer systems in bin handling systems in production environments. In table 2.3 different sources on storage optimization are summarized and their respective focus. Only Mueller, 2020 mentions buffer systems, but in an implementation study and does not mention the automation or optimization process. Studies into storage systems mainly focus on warehouse optimization. This does not mean principles from this field can not be applied on bin handling systems.

Source	Focus	
	Warehouse	Buffer systems
Quintanilla et al., 2015	**X**	
Yu and De Koster, 2013	**X**	
Yang et al., 2017	**X**	
Bahrami et al., 2019	**X**	
Park, 1987	**X**	
Mueller, 2020		**X**

Table 2.3: Different sources on storage optimization

This chapter aims to answer two subquestions, the first of which being: **what is the definition of buffering and how does this relate to warehousing?**. Hamada et al., 2006 defines a manufacturing buffer as a short term storage locating for semi-finished product in order to adjust for variation in the production process. The key term here is short. Buffers are often as small as possible since storage in the production environment is costly. Warehousing is for long term, bulk storage. Warehouses can take many different products over time. Also, an item is not always in stock. Shared storage strategies are therefore often useful in warehouses since they are large and with this strategy travel distances can be minimized. Buffers have only a small set of items they require to store, sometimes even only one. In order to not halt production these items will always be in stock. Here a dedicated storage strategy is mostly used. The storage part of a bin handling systems does not fit into the warehouse or buffer definition since more storage is required then just accommodate fluctuations in the production process. Therefore it is not clear what kind of storage strategy will be most useful from literature.

The second subquestion this chapter aims to answer is: **What are the design strategies present in existing bin handling systems?** In section 2.4 five distinctly different equipment types are defined by Kay, 2012. The equipment types can be translated to functions and these function can be used to determine possible function combinations in bin handling units. Further more, Mahadevan and Narendran, 1993 describes two types of storage or buffers in a bin handling system: central and local. Autonomous bin handling systems are novel and not a lot of research on them exists. Most existing bin handling systems use inflexible infrastructure or units such as conveyor belts. This thesis aims to find design solutions for flexible systems. Ye et al., 2018 shown us a method how different system functions can be integrated in one systems unit. This concept might also be useful for other units in the system. In section 2.2 the different field of operations research are explained. Since bin handling systems are part of the larger production process, this gives insight in interaction with production cells and other supporting systems. Operations research describes methods for optimizing different problems in operations. While not all proven to be relevant for this research, it is important to see what might have been useful. The generalized assignment methods is useful in the determination on how to shape the transportation and storage function of a bin handling system.

3 Methodology

To be able to accurately evaluate the impact of different design choices, the results have to be quantified. In this chapter the following subquestion will be answered: *How can a cost optimal solution be found using quantitative modeling?* Throughout this chapter, the steps necessary will be systematically explained to ultimately answer the subquestion.

3.1. Quantitative optimization

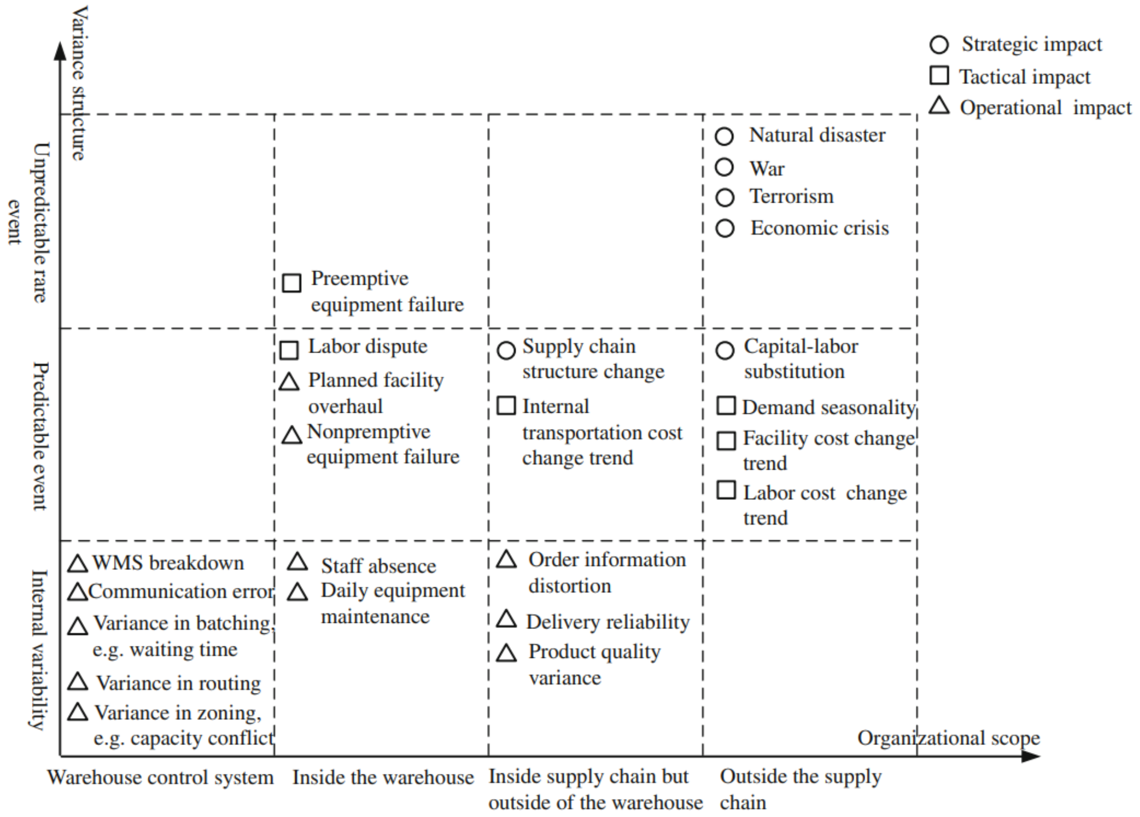
Quantitative methods are based on objective data that can be clearly communicated through statistics and numbers. This makes it measurable and comparable. This is why quantitative methods loan them selves quite good for optimization problems. Optimization is a broad term, describing the search for the best solution, as stated by Adby, 2013. The best solution however is subjective. One might be looking to minimize costs while another might be looking to maximize profit. The fundamentals for both problems remain the same. A system is described in a mathematical model and a cost function is identified. Next, a computational solver is used to find the optimal solution according to the cost function.

3.2. Deterministic and stochastic methods

Models can have a deterministic and a stochastic approach, Janssen et al., 2013. Deterministic models give the same output each time for the same inputs. Here the inputs are assumed as known values that do not change. This makes models easier to understand and implement. While some inputs can be safely assumed not to change, some inputs may have a spread in their value. In these cases the simplification of assuming a fixed value may lead to an inaccurate outcome of the model. When uncertainty in input values becomes undeniably influential a stochastic model approach can be adopted. Where deterministic models determine an exact solution, stochastic models predicts outcomes that account for certain levels of unpredictability or randomness, Kenton and James, 2021.

To describe a real world system as accurately as possible, a stochastic approach is the best option. A good example here is the throughput time of a product. In a deterministic model this could be defined as 10 minutes. In the real world this could be 10 minutes slower or faster for each individual product due to unforeseen reasons. This would mean the deterministic model is wrong. In a stochastic model the input could be given in the form of normal distribution around a mean of 10 minutes. This however results in more work in determining the required distribution and gives multiple outcomes while the errors introduced by the simplification done by the deterministic model may be negligible. The degree of deviation from the real world system indicates whether or not a deterministic model suffices. Since stochastic models are more complex, one might choose a deterministic model when able.

In field where more precision is required or more unpredictable variables are present we see more stochastic models. These fields include insurance, investing, statistics, biology and quantum physics, Kenton and James, 2021. Even though warehouse operations are much more predictable, there is some degree or uncertainty. Gong and de Koster, 2011 gives a review on the stochastic aspects of warehouse operations. In figure 3.1 he describes the sources of uncertainties in warehouse operations related to strategic, operational and tactical impact.



○ Strategic impact
□ Tactical impact
△ Operational impact
Variance structure
Unpredictable rare event
Predictable event
Internal variability
Organizational scope
Warehouse control system
Inside the warehouse
Inside supply chain but outside of the warehouse
Outside the supply chain
○ Natural disaster
○ War
○ Terrorism
○ Economic crisis
□ Preemptive equipment failure
□ Labor dispute
△ Planned facility overhaul
△ Nonpremptive equipment failure
○ Supply chain structure change
□ Internal transportation cost change trend
○ Capital-labor substitution
□ Demand seasonality
□ Facility cost change trend
□ Labor cost change trend
△ WMS breakdown
△ Communication error
△ Variance in batching e.g. waiting time
△ Variance in routing
△ Variance in zoning, e.g. capacity conflict
△ Staff absence
△ Daily equipment maintenance
△ Order information distortion
△ Delivery reliability
△ Product quality variance

Figure 3.1: Uncertainty sources of warehouse operations as stated by Gong and de Koster, 2011

3.3. Programming methods

To evaluate a deterministic problem, different programming methods could be used. Linear programming (LP) is the most common method since solving these equations is the least computationally heavy. LP is the maximization or minimization of a linear objective function, Karloff, 2008. Since stochastic methods often include a non-linear objective function, LP is more common in deterministic models. In LP the constraints of a system span a space in *n* dimensions. Next the objective function is assessed for the points in the spanned space and the minimum and maximum value can be found. For more complex problems dynamic programming (DP) can be used. In DP a problem is split up into multiple subproblems. Each subproblem is solved individually and then sub solutions are gathered and an optimal solution to the original problem is formulated. As stated by Sniedovich, 1991, this can only be done to objective functions that follow the formulations of a dynamic programming optimality equation. If an objective function is not separable as a DP problem and the objective function is of a higher order, non-linear programming methods are needed. The key principles are the same as in LP with the difference being that constraints can be of a higher order. This requires a more powerfull solver since the space spanned by the constraints might be more complex.

3.4. Mathematical models

Real world systems of any kind can be described using mathematics. Aarts, 2010 defines a mathematical model as: a simplified representation of certain aspects of a real system, capturing the essence of that system using mathematical concepts. Real world systems are very complex, therefore it is important to note here that a model is always a simplified representation. In figure 3.2 a graphical representation of the mathematical modeling cycle is depicted.

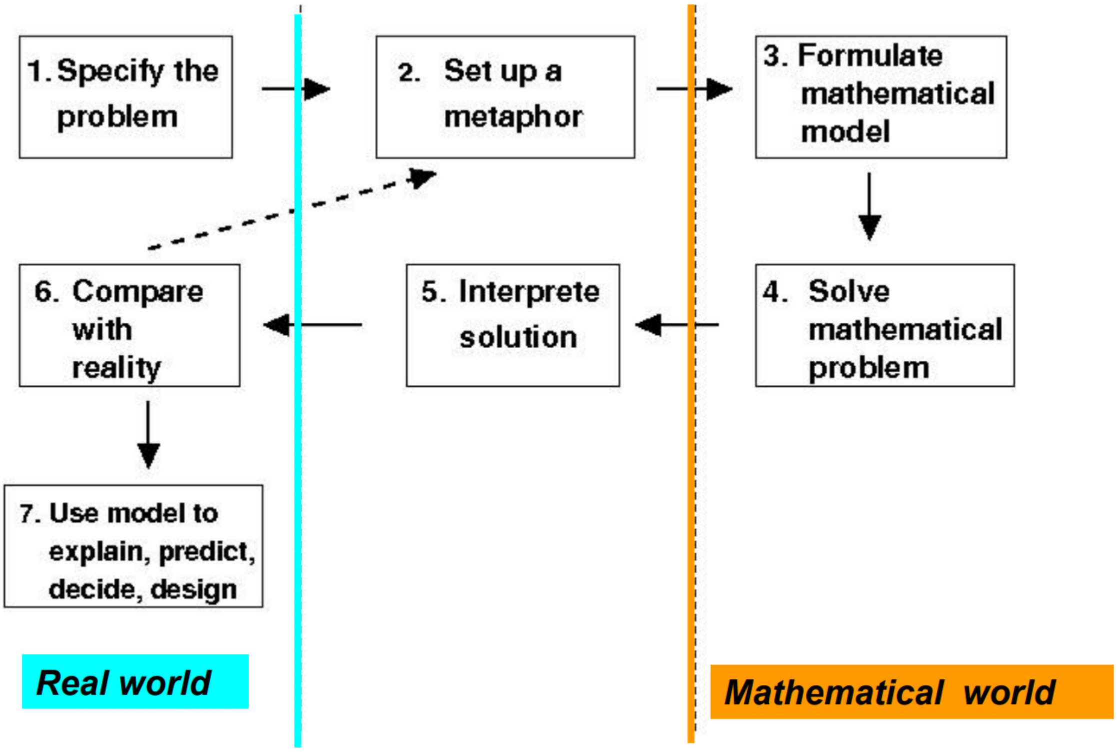


Figure 3.2: Mathematical modeling cycle as stated by Aarts, 2010

A mathematical system consists of the four following classes: decision variables, parameters, constraints, and and objective function, Duinkerken and Atasoy, 2020. A system is always bounded by rules. These could be physical rules, for example the inability to move due to bolted connections, or system imposed rules. All these rules are called constraints. Since real world system can vary a lot, each system has its own specific constraints. One constraint might be: the costs can not exceed the money available and another could be the production time equals one hour. Even though constraints are different, IBM, 2022 states all constraints could be categorized as one of the following:

- A not null constraint. Here a value or table index is prohibited from being zero.
- A unique constraint. Here a duplicate value is prohibited, for example when assigning supplier.
- A primary key constraint. Here relations between variables or tables are stated.
- a foreign key constraint. Here shared values between variables can be stated.
- a table check constraint. Here each variable can be prohibited from becoming lower or higher then a certain value.

Parameters are the inputs of the system. These are unchangeable values. A good example is the cost of raw material. These prices are fixed and the model can not decide on its value. The opposite of parameters are decision variables. Decision variables are free for the model to assign in order to find the best possible outcome. It may be cheaper to produce product A in January and product B in February instead of the other way around. The values of the decision variables after optimization can be used to tune the real world system. To determine the decision variables an objective function has to be defined. The objective function is the heart of the mathematical model. Here the decision variables are assigned a weight factor and the costs of changes in them are taken together. By minimizing or maximizing the objective function the best possible result is calculated.

3.5. Computational solver

To solve the model for the optimum value, a computational solver has to be used. If searches for the optimum value for a formula is the objective cell by changing other cells connected to it via constraints. First an initial solution is gathered by the solver. Next decision variables are altered in each iteration of the solver and evaluated with respect to the initial solution. By updating the solution if the found solution is better, the best solution is found after many iterations.

The solver used in this research is Gurobi Optimization for Python. Mathematical optimization solvers like Gurobi are able to efficiently process all available data and consider a large number of combinations of all relevant decision variables, constraints, and objectives Gurobi, 2021.

3.6. Verification and validation

Computer models can always contain errors. These errors may come from coding mistakes, unjustified simplifications or incorrect relations between components. As stated by Babuska and Oden, 2004 the question ”can computer predictions be used as a reliable bases for crucial decisions?” arises when using computer models. To assess the reliability and accuracy of a model it is crucial to verify and validate (V&V) it. Verification is the process of determining that a model implementation accurately represents the developer’s conceptual description of the model and the solution to the model as stated by Thacker et al., 2004. Here coding and calculation errors are identified and resolved. Code verification can be done through comparison of the results with analytical solutions to the mathematical model..

Validation assessment is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model, as stated by Thacker et al., 2004. While a mathematical model may be correct in the eyes of the developer, it might not accurately represent the real world system. To assess if the model is a good representation the results can be compared with the results of experiments. To validate the code and model one can also simulate impossible situations. For example a problem where there is demand but no supply. The model should give no result in these cases and break. If this is not true the model is not well defined. Sargent, 2010 suggests the following methods of validation: animation, comparison to other models, degenerate tests, event validity, extreme conditions tests and more. The methods of validation are numerous but depend on the resources and data available. In figure 3.3 the verification and validation steps are illustrated in the model design process.

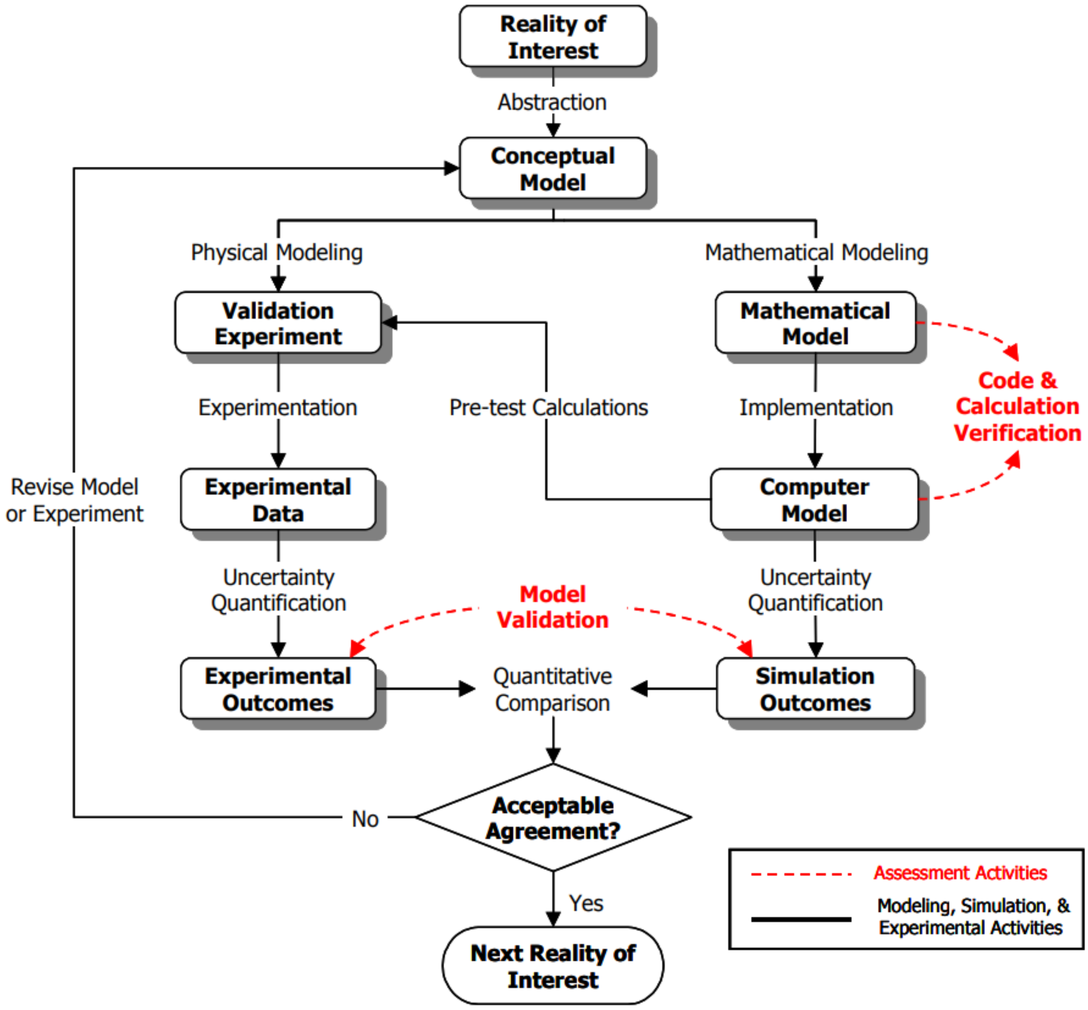


Figure 3.3: Model verification and validation in the design process, Thacker et al., 2004

3.7. Performance indicators

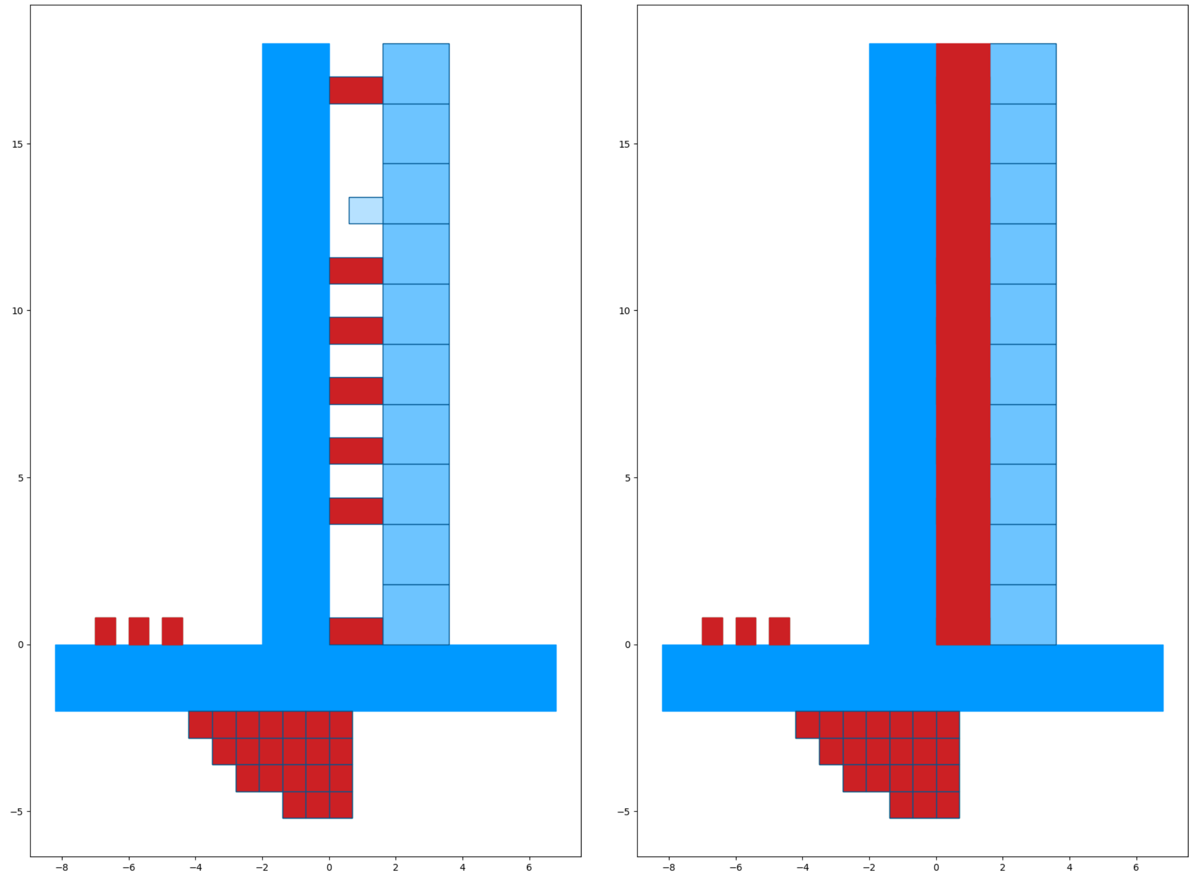
To evaluate the performance of a system, performance indicators need to be specified. In this section the key performance indicators (KPIs) of this research are introduced. Information on what the KPIs are, why they are relevant and how they are measured is given.

3.7.1. BOM costs

Implementing a bin handling system comes with different costs sources. One of which is the bill of material (BOM) costs. Other cost factors are indicated in section 3.8. Each unit in a system had a price. The BOM is the total price of all components required for a certain unit. More complex units often have a higher BOM. The total BOM of a system is therefore not only dictated by the number of units, but also by the type of units in the system. BOM is one of the KPIs used to evaluate the performance of the system.

3.7.2. Floorspace

The second KPI is floorspace. Each system takes up a certain amount of space. Floorspace can be presented in two ways: individual unit footprint and wasted space system footprint. When assuming individual unit footprint, the footprint of each unit is added to determine the total footprint of the system. This approach does not consider space between units. This space can be considered as 'wasted'. In figure 3.5 these different methods are depicted. In red the total system footprint is indicated. Walkways and production cells are indicated in blue. A notable difference in total footprint can be seen here. Since this wasted space can not be ignored, this thesis considers the method of including wasted space as indicated in figure 3.4b.



(a) Individual footprint

(b) Wasted space footprint

Figure 3.4: Example of floorspace definitions

3.7.3. Demand capabilities

The last KPI considered in this thesis is the ability to cope with increasing demand. This is a KPI indicating how the system had to be expanded in order to cope with an increasing demand. In this thesis, the demand of a system is based on the takt time of a product and a period of one shift. One shift is defined as 8 hours of non stop production. Takt time indicates the time it takes for one product to be produced. With a shorter takt time, more product will be produced in one shift, thus increasing the demand for each production cell. Three main factors are identified which increase costs and floorspace. First, required storage space will increase since more semi-finished product need to be stored during the shift. Next, the number of jobs for the AGVs will increase. This in turn will result in more AGVs required in the system. Lastly, the (de)stacker capabilities. The (de)stackers must keep up with the demand of the cells.

3.8. Optimization objectives

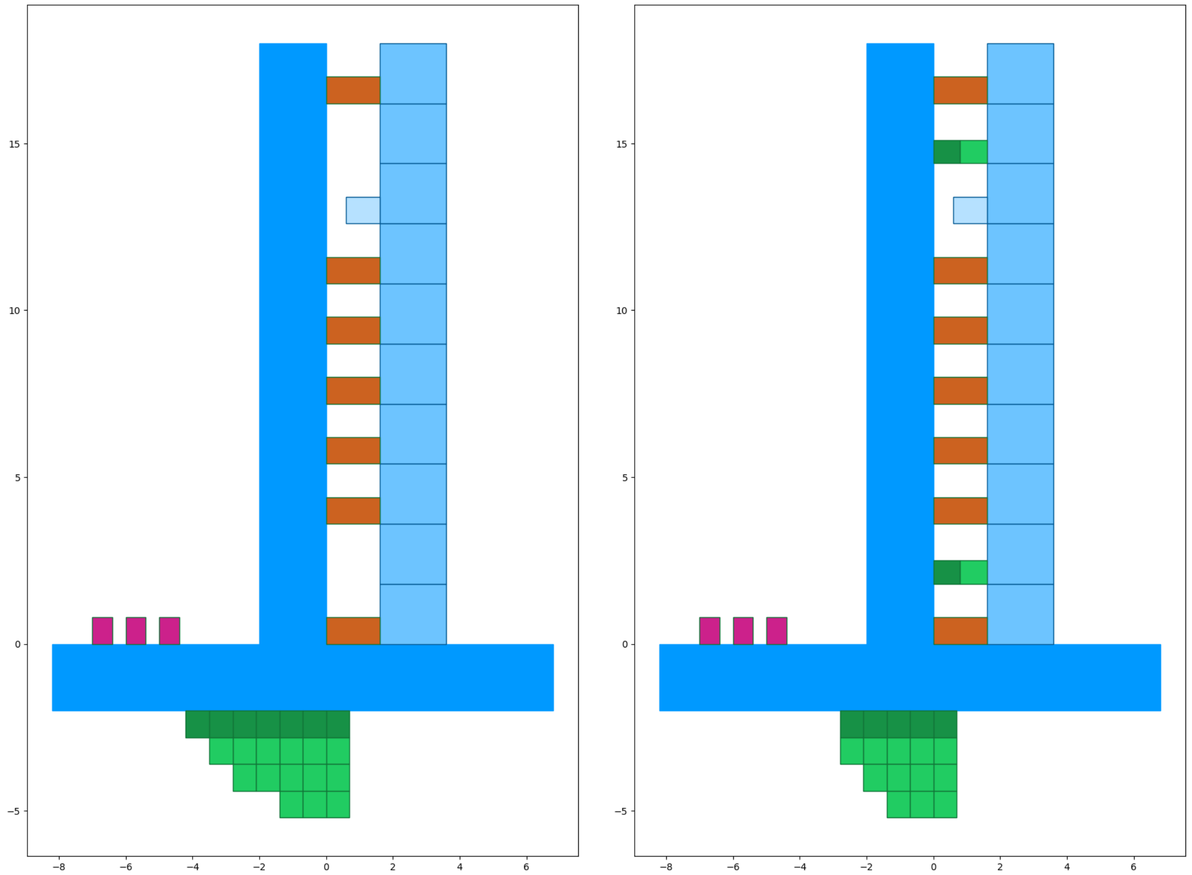
When a mathematical model as discussed in 3.4 is implemented, an optimization objective has to be identified. Depending on what this objective is, the solver will prioritize certain decisions above others. Below the three different optimization objectives of our model are discussed. These different objectives are used to analyse the decision making logic of the model and see the difference in performance for different optimal solutions.

3.8.1. BOM

The first objective is the BOM. Focusing solely on the costs of units, the model will choose the cheapest option possible. Floorspace optimization will be neglected due to it not being in the objective function. It might be cheaper to place units outside of the wasted area since there is more space outside of it. Depending of where a company is located, floorspace might be very cheap, or very costly. When floorspace is very cheap, only optimizing for the BOM might be interesting.

3.8.2. Floorspace

As stated, depending on the location of a company floorspace might be very cheap or expensive. In case of the latter, optimizing for floorspace only could be of interest. Placement of the units now plays a larger role. In case of floorspace optimization the model will most likely choose to place units in the wasted space area in order to save space. This will be done even tough other costs might increase. In figure 3.5 the difference in unit placement is visualized. Storage units are indicated in green. In figure 3.5a an example of a possible BOM optimal solution is illustrated and in figure 3.5b an example of a possible floorspace optimal solution. Here it can be seen how the model proposes a location for the storage units based on the floorspace and wasted space costs.



(a) BOM optimal placement

(b) Floorspace optimal placement

Figure 3.5: Visualisation of floorspace optimization objective

3.8.3. Total costs

Lastly, the model should also be able to optimize the total system costs. These costs consist of four different factors:

1. BOM costs
2. Assembly costs
3. Installation costs
4. Floorspace costs

BOM, assembly and installation costs all are directly tied to the unit amount. Each different unit in a system comes in as components and need to be assembled and installed. More units means more assembly and installation costs. Floorspace costs are based on the total system footprint, including wasted space. To include square meters into the total costs and conversion has to be done. By including the write-off of a square meter of floorspace, the footprint of the system can be included in the costs. However, write-off is time based. That is why a time span has to be included in the total costs. Therefore the total costs are defined as: **the costs of a system over it's lifespan.** The depreciation period of the units is the leading indicator for lifespan of the system. For instance: if the units of a system have a depreciation period of ten years and the floorspace depreciation is 1 euro per day per square meter, floorspace costs will be 3650 euro per square meter. Optimizing for total costs will result in a balanced approach, placing units in the wasted space area only if the total cost decreases.

3.9. Generative design

As with all production environments, layout design is important for optimal performance. With layout design, a multitude of factors should be considered such as facility shape, manufacturing systems and material handling systems. As stated by Drira et al., 2006, there are even more factors which are depicted in figure 3.6. Considering all these factors is difficult when analytically constructing a layout and results rely heavily on the experience and creativity of the involved planning experts, Süße and Putz, 2021. Another way to develop layouts is via generative design methods. Generative design enables a much larger solution space and can provide the user with a visual design proposition.

When looking at figure 3.6 by Drira et al., 2006, only one aspect of all factors stated is relevant to our system: the material handling system. The facility shape is assumed as predefined and the manufacturing system layout is considered as input as stated in section 1.5. The degrees of freedom for a generative design method is therefore limited in this thesis. Since floorspace is an KPI considered in this research, generative design is still helpful. Placement of units is chosen by the model, and a generative design method can provide an example of the generated layout including the bin handling system.

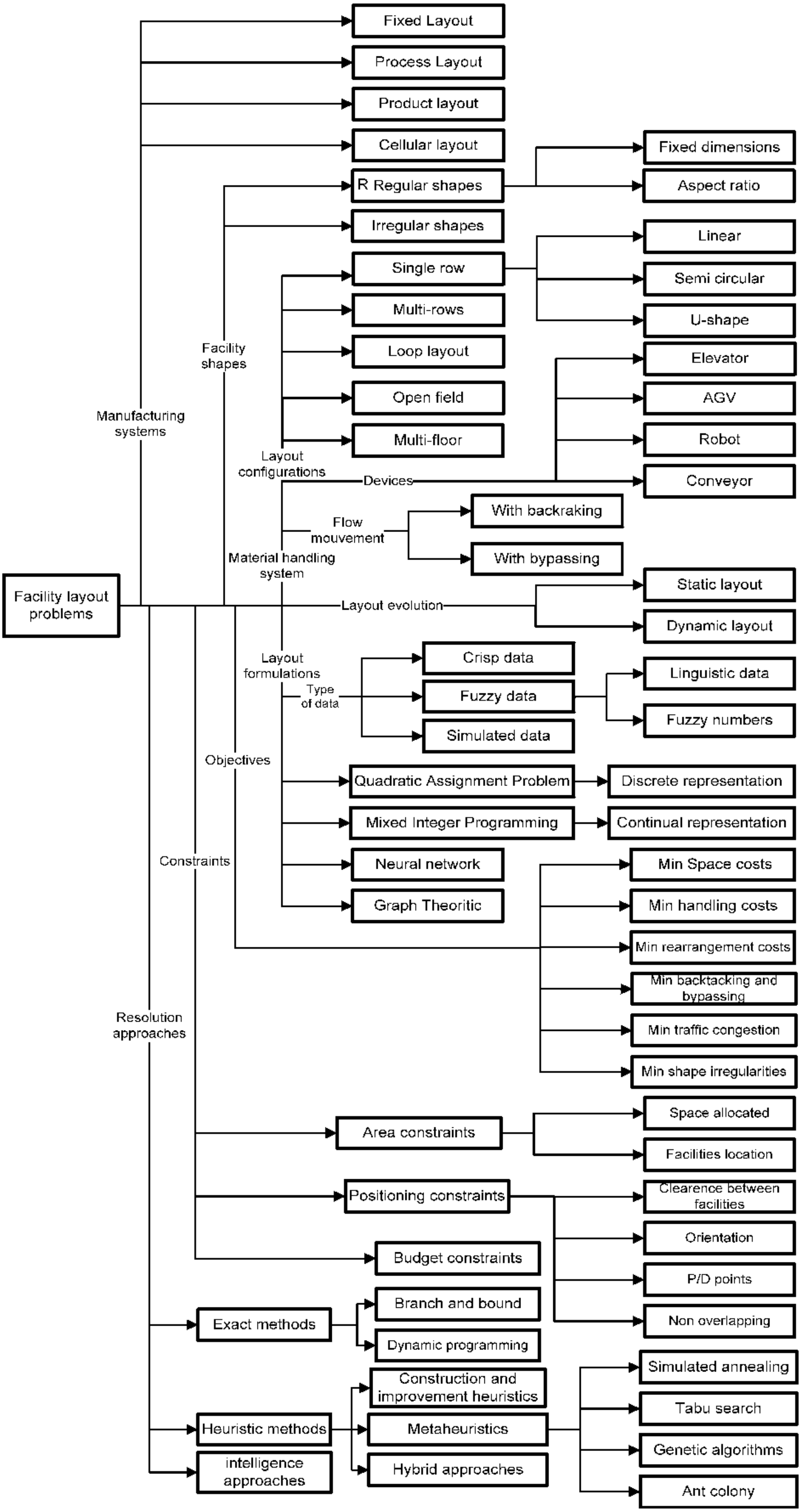


Figure 3.6: Facility layout consideration factors as stated by Drira et al., 2006

3.10. Chapter summary

In this chapter an answer is formulated for the subquestion: **How can an optimal solution be found using quantitative modeling?**. First the use of stochastic modeling methods is considered. While stochastic methods often have a higher accuracy, the question can be asked if this is necessary. In fully autonomous environments predictability is high. Add to this the fact stochastic models are more computationally heavy, a deterministic approach seems appropriate. The deterministic model can then be checked for critical demand and to what degree fluctuations can be absorbed. In order to construct a model, the programming method should be identified. Multiple methods are defined in this chapter. With the deterministic approach, a linear programming method seems to be appropriate.

To further quantify a real world bin handling system, a mathematical model of the system can be constructed. This is done through interpretation of the system in terms of formulas and constraints. To achieve the desired level of accuracy of a model that reflects reality, it might be necessary to repeat the development process several times. To solve the mathematical model a computational solver is used. In this thesis, the solver Gurobi is chosen since it is optimised for linear deterministic models. Once a model is constructed, it needs to be verified and validated. Verification and validation are in essence described by the the questions: ’is the model right’ and ’is it the right model’.

To find an ’optimal solution’, the definition of optimal should be clear. In this chapter three different optimal solutions are identified: the bill of material, system footprint and total costs. The bill of material is easily quantifiable since it is already expressed in term of costs. Floorspace however is expressed in square meters. Also, a bin handling system might use more space then just the individual footprint of each unit. Therefore the system footprint is defined as all the space required for the system to operate, and the space wasted by unit placement. Wasted space refers to areas where unit placement or movement restricts other activities from taking place, resulting in unused or unusable space. This total footprint can then be quantified as costs by relating it to the write-off costs per square meter. The model can also be used to find an equilibrium between bill of material and footprint optimal solutions. This represents the total costs optimisation.

II

System identification

4

System overview

To accurately construct a mathematical model, the system must be clear. In this chapter the modules of an existing bin handling systems will be identified, as well as the operational environment. The system described is based on a real world case. Eventually, this chapter will answer the subquestion: *How is a bin warehouse defined and how do the costs relate to the system?*

4.1. System definition

Our system consists of three stages: the introduction stage, the storage stage and the production stage. The introduction stage is the stage where items and stacks of items enter the system. Here an operator is instructed to place a stack of items on single payload exchange platform (SPEL) *X* by LES. LES then sends a signal to the storage system telling it there is a stack of items introduced and waiting at SPEL *X*. From now on the stack of bins is inside our system and is what we call the supply. The amount of supplies introduced is regulated by LES and depends on the quantities being produced. Our systems only considers the items after they are introduced. This means the speed of the operator introducing them is not individually considered. Rather it is summarized by a supply parameter which can be varied later on in testing. Opposite to the supply is the demand. The demand of our system is determined by the production stage. Each individual production cell in the production line uses products. One might use large, bulky products while another uses small products. These products all come in the same standardized bins. Since the measurements of these bins are standardized, more small products are supplied in one bin than large products. This makes the demand of different production cells larger or smaller.

4.1.1. Second generation particle sensor

To fully understand the system, we also have to be informed on its environment. Our bin handling system is integrated in the second generation particle sensor (SGPS) production line. The production process consists of 33 steps. Each step is carried out by a production cell with a different function. The layout of the production line is illustrated in figure 4.1. production cells, walkways and liquid reservoirs are depicted in blue. The (de)stacker cabinets (DSCAB) are depicted in orange, the SPELs in red and the multi bin cabinets (MBC) in green. Dark green indicated an MBC master and lighter green an MBC HBT.

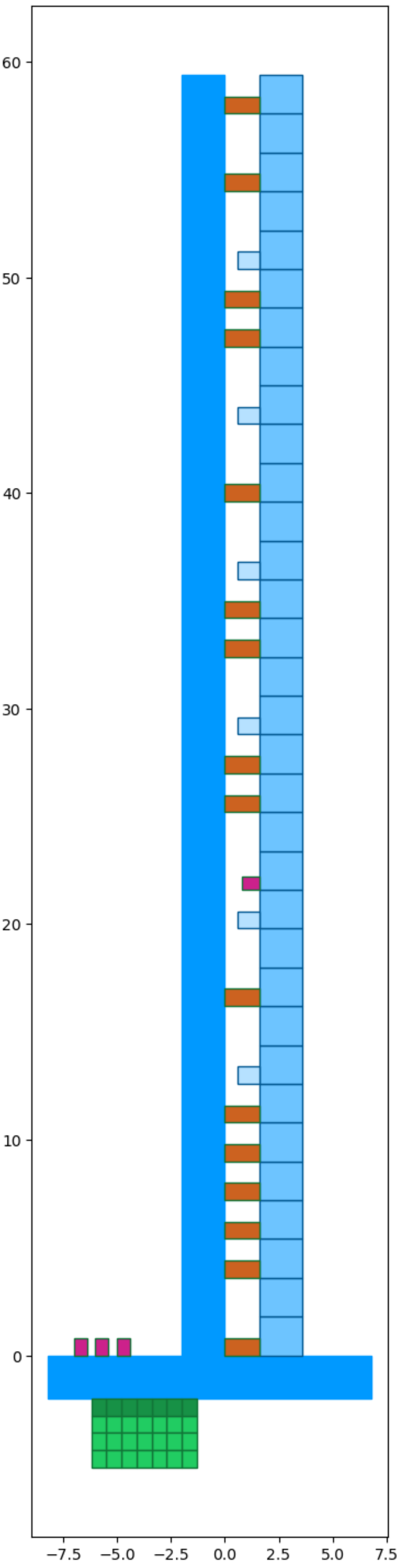


Figure 4.1: Illustration of production environment SGPS line

The Takt-time of the SGPS production is 60 seconds. The bin handling system is required to accommodate this at all times, meaning enough buffer and supply should be in the system to not slow down production. Also, components should enter the cells at the right time. This is currently done through the placement of an DSCAB in front of each of the individual production cells. Through the DSCABs a cell can be supplied with one stack of bins at a time, and de-stack these bins itself. This means the AGV does not have to make a trip to the cell each time a bin is empty. The empty bins are also stacked by the DSCAB, meaning the AGV can pick up multiple empty bins at once.

Even though all cells have the same throughput, supply requirements of each cell are different. This is due to the difference in size of the parts required in the operation done by the cell. Parts are delivered in bins. These bins are lined with a mold ensuring the right orientation of each part. Since parts are of different sizes, the number of parts per bin differs. The demand of a cell is therefore defined as bins per shift, instead of parts per shift.

4.1.2. Bin handling units

As stated in section 1.3, bin handling systems can consist of any number of different units. The system in this case study consists of four distinctly different units. The four units are the SPEL, DSCAB, MBC and AGV. Below each of the units are further elaborated.

Stand-alone Payload Exchange Location

The Stand-alone Payload Exchange Location (SPEL) is a stand alone or coupled unit used for interaction with operators or as connection between AGVs and external units. Since a SPEL is accessible from all sides, operators can place or remove bins easily. It consists of one payload location, but can be coupled with multiple SPELs to create longer lanes. Since they are accessible from both sides by an AGV both a last in first out (LIFO) or first in first out (FIFO) strategy can be used. In figure 4.2 the schematic configuration of a SPEL is depicted. In figure 4.3 a render of the SPEL can be seen. The SPEL is capable of fulfilling the following functions within the system:

- Item introduction / extraction (operators)
- Storage

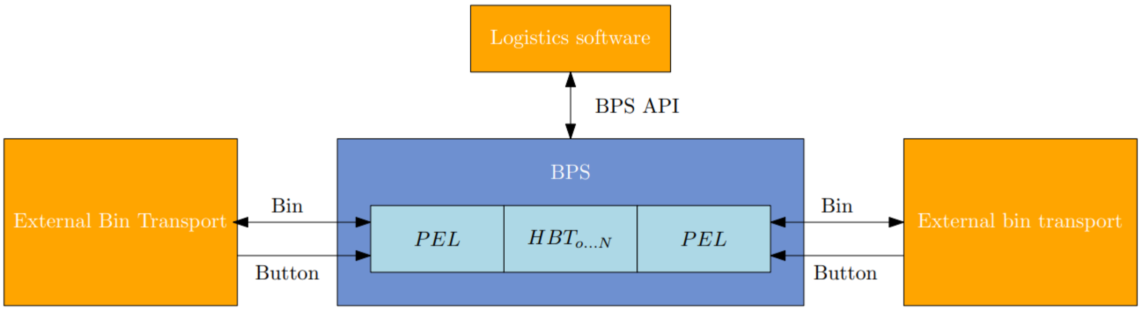


Figure 4.2: Schematic overview of a SPEL

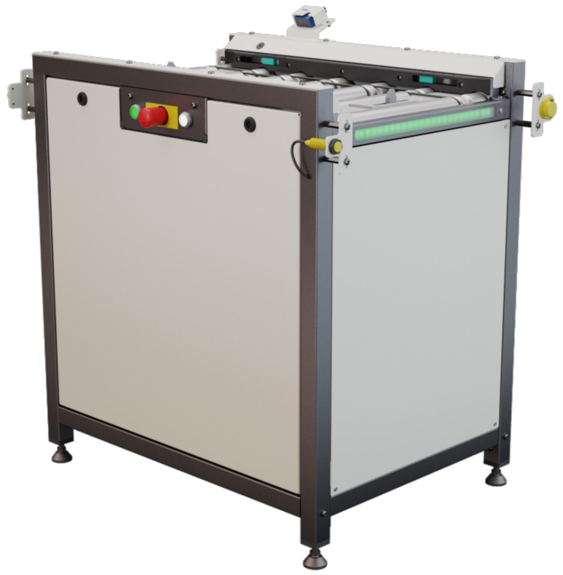


Figure 4.3: Render of a SPEL unit

(De)Stacking Cabinet

Transporting a stack of items instead of single bins drastically reduces the amount of AGV trips needed. The bins arrive in the systems already stacked from the warehouse. However, the production cells need single bins in order to access all the components. This means between the transportation and production, a destacking unit is needed. The (De)stacking cabinet (DSCAB) is developed for this task. It is not only capable of destacking full bins, it can also stack the empty bins. In a DSCAB the are two payload exchange locations (PELs) each with a (de)stacking module(DSM) above or below it as can be seen in figure 4.4. The PEL accepts a stack of items and carries is to the vertical bin transport (VBT) module. Here the stack is elevated and delivered to the DSM. The DSM can then hold the stack and return the bins one by one. The reverse of this process can also be done with empty bins in order to transport stacks of empty bins back to the warehouse. In figure 4.5 a render of an DSCAB is depicted. The DSCAB is capable of fulfilling the following functions within the system:

- Item introduction / extraction (production cells)

- Storage

- Stacking

- Destacking

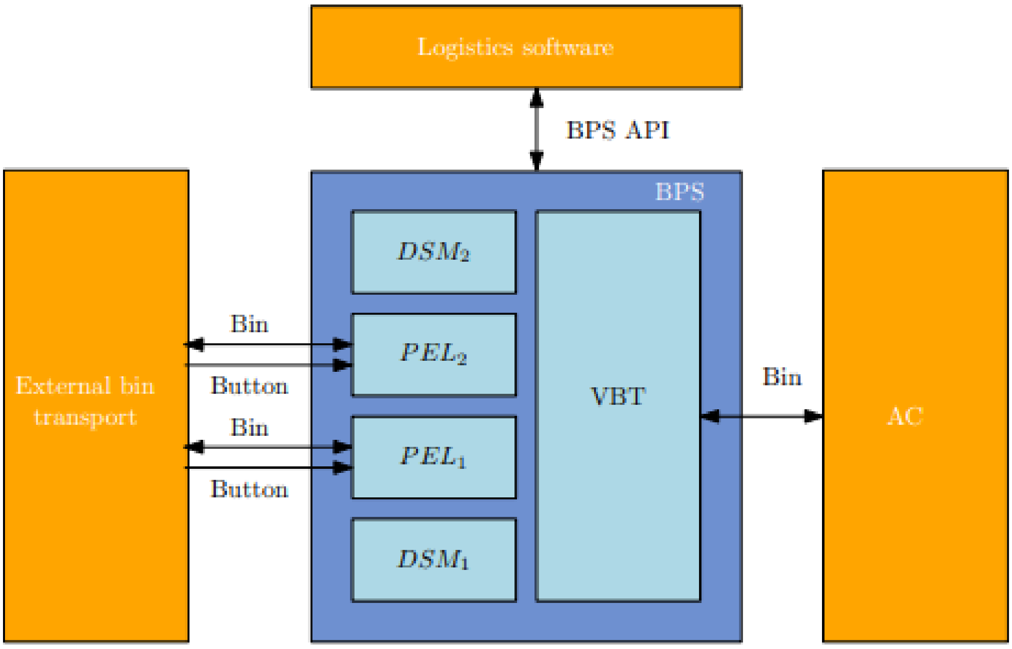


Figure 4.4: Schematic overview of a DSCAB

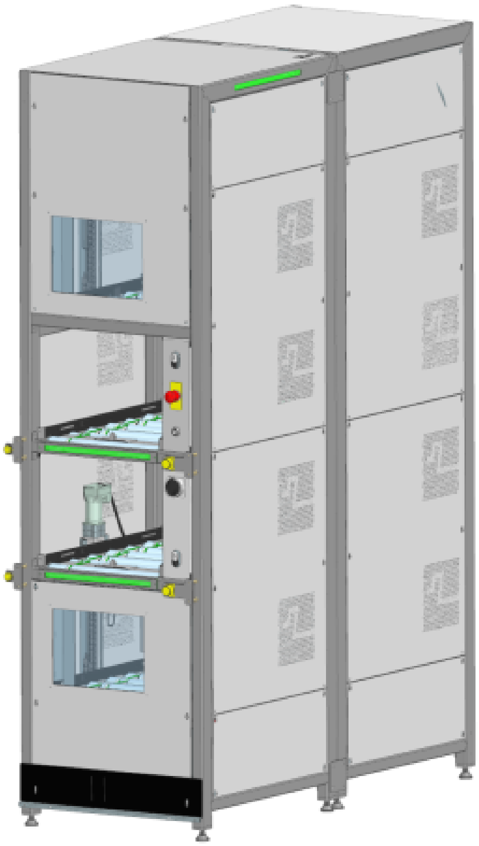


Figure 4.5: Render of a DSCAB unit

Automated guided vehicle

Transport between the introduction stations, buffer cabinets and production cells is handled by automated guided vehicles (AGVs). These AGVs are capable of carrying one stack of items per trip. While still guided by a logistic execution system (LES), the AGVs drive fully autonomous. They constantly scan for obstacles and slowdown or stop when identifying one. The LES system gives the commands to the AGVs where to pickup a product and where to deliver it. In table 4.1 Once arrived at a payload exchange location (PEL), the AGV scans for the PEL orientation and aligns itself. The AGV is capable of fulfilling the following functions within the system:

- Item transportation

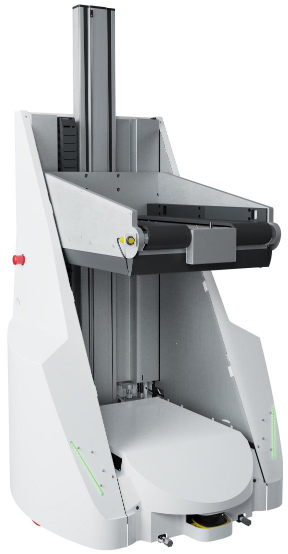


Figure 4.6: Render of an AGV unit

Action	
AGV depart time (s)	5
AGV approach time (s)	15
AGV (un)load time (s)	5
Average AGV speed (m/s)	0.8
Availability per AGV (s/hour)	3600

Table 4.1: AGV properties

Multi bin cabinet

The bulk of the storage space is provided by the multi bin cabinets (MBCs). These units provide four unique storage locations (USLs) above each other. Since production environments can vary in available space and demand, these units are modular in nature. Meaning multiple MBCs can be placed behind each other in order to enlarge the size of each USL provided. To achieve this, three versions of the MBC are developed: The MBC master, MBC Horizontal Bin Transport (HBT), and MBC slave. MBC master units will always be placed first in a row. This is due to the AGV docking capabilities the master cabinet has. The MBC HBT units are designed for the expansion of the USLs. Since they do not have AGV docking capabilities they need an MBC master to operate. This lack of docking capabilities also reduces their price. When access to the goods from the back of the row is required, an MBC slave can be used. Slaves also have the docking capabilities the master has but lack the internal communication capabilities since this is already done by the master. When using filling strategies such as FIFO, a slave is required. In figure 4.7 a schematic overview of an MBC configuration is depicted. The scheme is similar to that of the SPEL with the difference being the capacity is four times as big in the same floor space. Finally, in figure 4.8, a render of an MBC configuration is depicted. The MBC is capable of fulfilling the following functions within the system:

- Storage

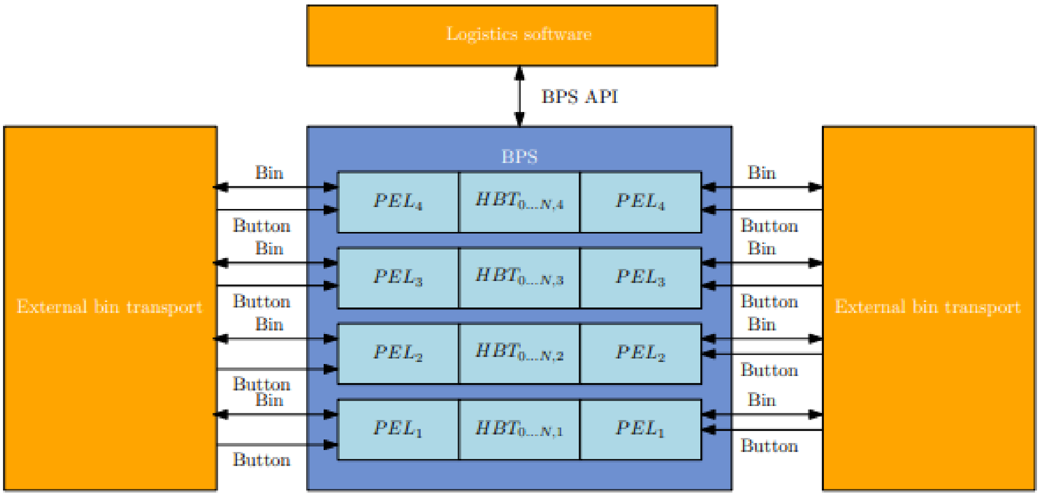


Figure 4.7: Schematic overview of an MBC configuration

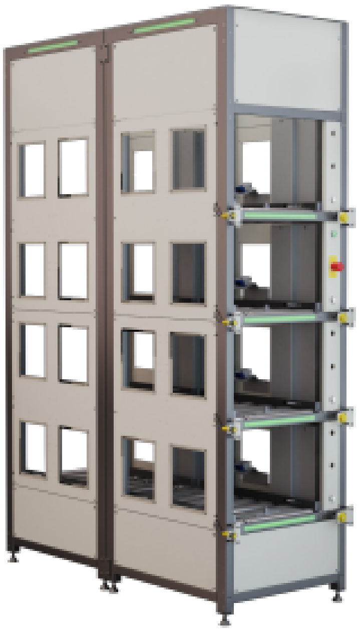


Figure 4.8: Render of an MBC configuration of a master and a HBT

4.2. Costs

Due to confidentiality, real numbers cannot be given in this case study. It is however important to have a sense of how the different costs relate to each other. Therefore costs will be depicted relative to each other as can be seen in table 4.2. For example, an MBC master costs almost twice as much as an MBC HBT. The costs of using the bin handling system consists of different aspects as stated in section 3.8. The development costs are not considered in the implementation since they are shared over the entire program and all later uses. The costs considered are the following:

- Bill of materials
- Installation costs
- Assembly costs
- Floorspace costs

Bill of material

The bill of material (BOM) consist of all the costs of all parts needed to create the units. Each unit has a different BOM. Since a SPEL uses less parts, it is cheaper to produce then for instance an MBC master. In table 4.2 the BOM of the different units is depicted. Notably is the difference in BOM between an MBC master and an MBC HBT. This is due to the docking and communication capabilities of the master unit. The totality of the BOM costs can be found by multiplying the number of units with their BOM costs. The units needed are expected to be new, meaning they all need to be bought and there are no 'in stock' and all units require the full BOM cost to implement.

	SPEL	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGV**
BOM single unit	35%	100%	55%	95%	250%	220%

Table 4.2: BOM costs per unit relative to the costs of the MBC Master

Installation and assembly costs

Since the buffer units arrive as parts, they need to be assembled and installed in the right location. To do this a mechanic is required. The costs of assembling and installing a unit can be determined through the time it costs to assemble and install and the hourly rate of a mechanic. The hourly rate of the mechanic can be referred to as the full time equivalent (FTE) of a mechanic. An FTE is the measuring unit of workhours of a certain capacity and can be related to a price. In table 4.3 the amount of (day's worth) FTEs it takes to assemble and install each unit is depicted.

	SPEL	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGV**
Assembly time	1	1,5	1,25	1,5	1,5	2
Installation time	0,25	0,5	0,4	0,5	0,5	0,5

Table 4.3: Assembly and installation time (in days for one FTE)

Floor space costs

Floor space within a factory is not free. Occupying space means other production lines have less space to be installed, eventually leading to the need for a larger production hall. This is why floor space has to be depreciated. The price of floor space is indicated by euros per m^2 per unit of time. This means this costs source is time dependent. In table 4.4 the dimensions of the different units are depicted. Aside from the units, the wasted space as explained in section 3.7.2 should also be defined as well as other external units such as liquid reservoirs.

4.2.1. Maintenance access

As mentioned in section 3.7.2, there are multiple methods of defining system footprint. Since placement of units might hinder placement of other equipment in some places, space between units is seen as wasted. While true, not all usage is hindered. Short term usage of these places can still occur. There is one other factor in this wasted space: maintenance hatches. One part of each cell is outfitted with a maintenance hatch. In figure 4.9 a close up of a single production cell is given. The cell is shown in blue. The different shades of green each indicate a possible place for an MBC unit. The maintenance hatch is shown in red. A can be seen, placement of MBCs in two of the four locations will obstruct access to the maintenance hatch. These locations are therefore not available for the bin handling system. This means placement of bin handling units is not the only factor restricting use of this wasted space. However, since the bin handling system does contribute, it is seen as part of the system footprint.

	SPEL	MBC Master	MBC HBT	MBc Slave	DSCAB	AGV
Width(m)	0,7	0,7	0,7	0,7	0,7	1,3
Depth (m)	0,8	0,9	0,8	0,9	1,3	1,2
Footprint (m^2)	0,56	0,63	0,56	0,63	0,91	1,56

Table 4.4: Dimensions of the different units

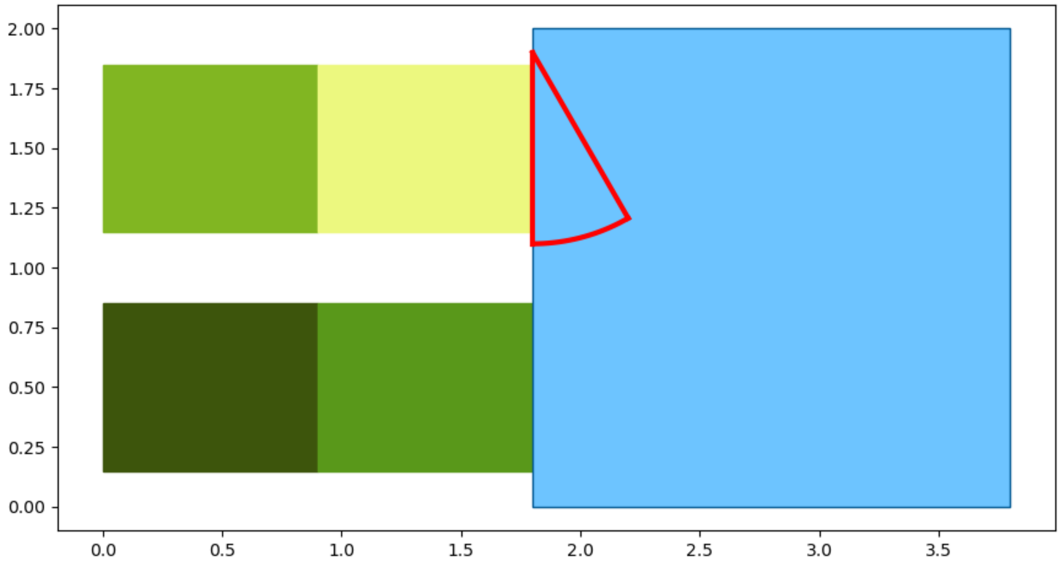


Figure 4.9: Zoom in of a production cell, possible unit placement and maintenance hatch

4.3. Chapter summary

In this chapter an existing bin handling system is explored in order to find an answer on the subquestion: ”How is a bin warehouse defined and how do the costs relate to the system?”. While the case study system consists of four distinctly different units, does not mean this is required by all bin handling systems. However, though these four units the different required functions of a bin handling system can be defined:

- Item introduction / extraction (operators)
- Item introduction / extraction (production cells)
- Storage
- Stacking
- Destacking
- Item transportation

The goal of an autonomous bin handling system is supplying the production line without human interaction. To reach this goal the system should, at minimum, be able to autonomously carry out these functions. The number and type of units required by the system dictate the costs in four ways as discussed in section 4.2. More, cheaper units might negatively impact the system footprint and increase the floorspace costs. Less, more expensive units might increase total BOM costs.

5

System analysis

The bin handling system in our case study is already in use. The implementation however is done through a quick calculation rather than a digital model and optimization strategy. In this chapter the following subquestion will be answered: *How does a quantitative cost optimal solution of a bin handling system relate to an analytical solution?* First the current analytical strategy will analyzed and the costs of implementing through this method will be determined. Next a quantitative optimization will be done using a mathematical model of the current system.

5.1. Dedicated storage and MBC usage

An MBC cluster can consist of multiple units. The first unit is always a master MBC with 4 rows and only one location deep. As said earlier, each row can only hold one type of product at once due to the dedicated storage restriction. We call a location with one specific product a unique storage location (USL). This means the amount of HBTs behind a master will expand the size of an USL and will not add more. In the strategy that is employed calculations are done with the assumption an MBC cluster always has one master and three HBTs. This creates four USLs with a size of four stacks of bins each.

5.1.1. Production demand

The production of the SGPS sensor consists of 32 individual steps. From these steps, 15 of them require a bin with components. In the table 5.1 these steps are numbered 1 through 15. Since components have different dimensions, two types of bins are used: 50 mm high and 75 mm high. This difference in height means a stack of 50 mm bins contains 7 bins while a stack 75 mm bins only contains 5. This is due to the clearance height of a MBC. In the table the amount of bins, and what type of bin, needed by a production cell is indicated.

Production Step	Bins		Total stacks required	MBC rows required	Unique storage locations required	MBCs required
	50 mm	75 mm				
1	5		1	0,25	1	0,25
2	11		2	0,5	1	0,25
3	41		6	1,5	2	0,5
4	13		2	0,5	1	0,25
5	13		2	0,5	1	0,25
6		82	17	4,25	5	1,25
7	82		12	3	3	0,75
8	20		3	0,75	1	0,25
9	31		5	1,25	2	0,5
10	25		4	1	1	0,25
11	49		7	1,75	2	0,5
12	4		1	0,25	1	0,25
13	4		1	0,25	1	0,25
14	6		1	0,25	1	0,25
15	16		3	0,75	1	0,25
Total	**320**	**82**	**67**		**24**	**6**

Table 5.1: Calculation sheet of amount of MBCs required

In the calculations there are many cases of rounding up. The best example of this can be seen in the data of production step one. There only one stack location is required. As stated in section 5.1, the current strategy only considers USLs with a capacity of 4 stack locations. This results in the need for an overly spacious storage location for the product demanded by production step one. In this USL there will be three empty stack locations. Due to the dedicated storage strategy these spaces can not be used for other products.

5.2. AGV, DSCAB and SPEL usage

The amount of AGVs needed for the buffer system is calculated with the inputs shown in table 5.2. In this calculation the assumption is made that charging has no impact on availability of the AGV. As for the jobs to be carried out by the AGV the following definition is used. A job is considered to be the actions of approaching. loading, departing, traveling, approaching and unloading at the target destination. This means bringing and returning to the starting position equals two jobs. After calculations the usage of AGVs is 3160 seconds per hour. With an AGV being available for 3600 second each hour, one unit will suffice.

Inputs	
AGV depart time (s)	5
AGV approach time (s)	15
AGV (un)load time (s)	5
Average AGV speed (m/s)	0.8
Availability per AGV (s/hour)	3600

Table 5.2: AGV inputs

The usage of DSCABs is bound by mechanical constraints of the production cells. Production cells require a single bin to be placed near the automated robot arm. Since the bins travel through the environment in stacks, they have to be destacked. The production cell itself is not capable of this, which means an additional unit is required to carry out this job. A DSCAB is able to accept a stack and return a single bin at a time. Placing one DSCAB in a single central location for destacking will results in more jobs for the AGVs. To still be able to deliver a stack of bins to a production cell, each cell has its own DSCAB. The DSCABs are then also able to stack up the empty bins to be picked up by the AGV again. From table 5.1 we see there are fifteen production cell using bins, and thus needing DSCABs.

As stated in section 4.1.2, the required amount of SPELs depends on numerous factors, many of them outside of our scope. In the current state configuration a total of three SPELs are used.

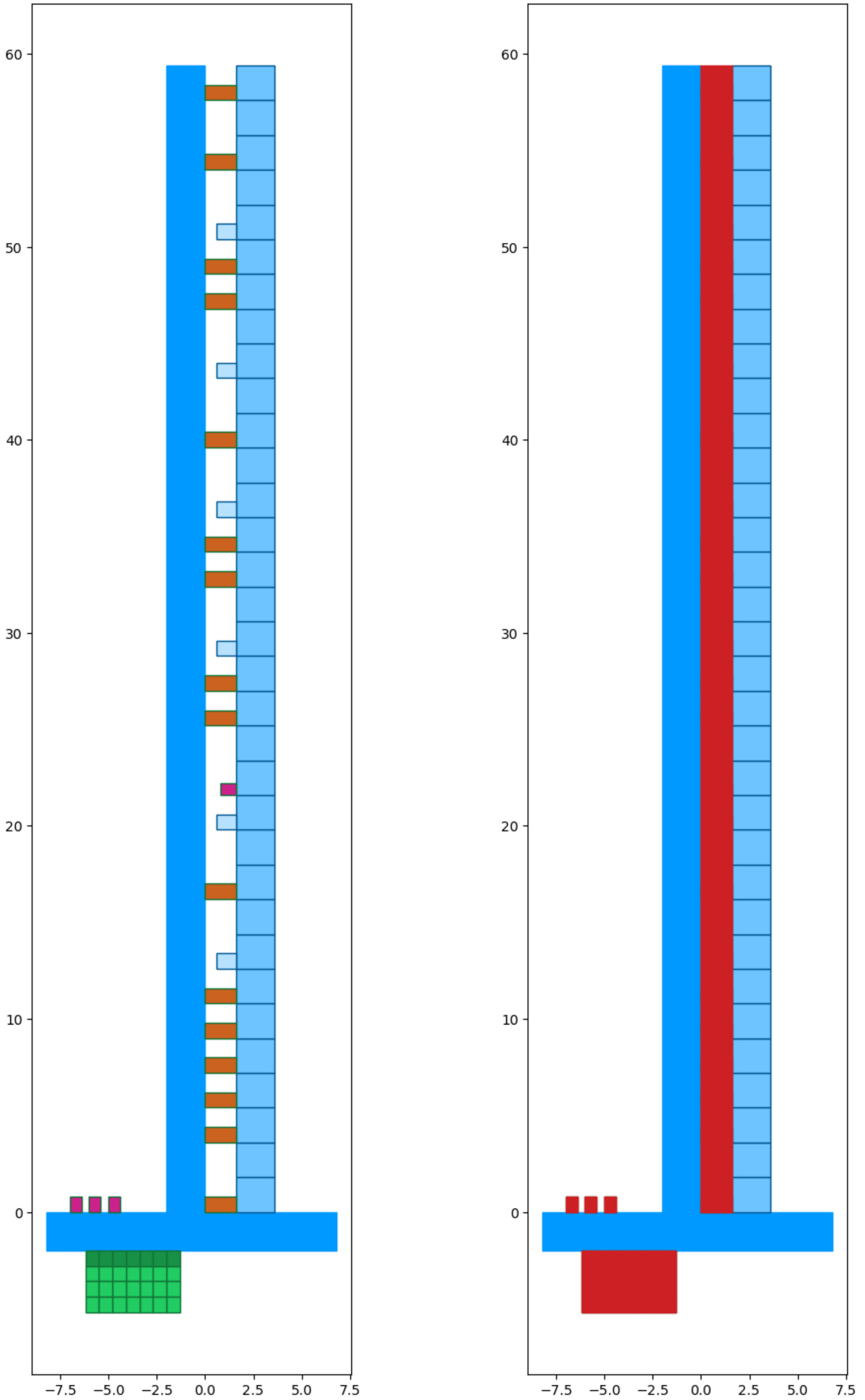
5.3. Current unit-assignment costs

Combining the data and calculations from the previous sections we can find the total number of units required, as well as the system footprint. This is depicted in table 5.3. In figure 5.1 the two environments are plotted by our model.

	SPEL	MBC Master	MBC HBT	MBC Slave	DSCAB	AGVs
Number of units	4	7	21	0	16	1

	Total	Wasted space	Individual
Square meters	114,1	71,6	42,5

Table 5.3: Results analytical solution



(a) Environment

(b) System footprint

Figure 5.1: Analytical solution to system implementation

5.4. Mathematical model

The current state of the bin handling system indicated the constraints on which it is bound. For example the dedicated storage constraint or the DSCAB required at each production cell. In the coming section a mathematical model of the current state will be developed as stated in section 3.4. Since our sets are dependent on parameters used as inputs the parameters are introduced first.

5.4.1. Parameters

Parameters are the inputs for our model. Parameters can be divided into two types. The first of which being inputs by the user. These parameters indicate things such as demand, costs per unit, system constraints, etc. The second being parameters calculated from inputs by the user. These are called parameters since they do not change through the iteration of the model solver. The user could be asked to do the calculations of these parameters themselves but doing them in the model makes outcomes more reliable. Below the used parameters are described.

Parameters	Info
UnitNames	Text array with names of the buffer units
Units_init	Array of initial amount of buffer units
BOM	Array of Bill of Material of each buffer unit
AssemblyTime	Array of assembly time of each buffer unit
InstallationTime	Array of installation time of each buffer unit
Footprint	Array of footprint of each buffer unit
BufferTime	Integer of for how long the system should operate without resupply
TaktTime	Integer of takt time of a finished product
D	Array of the demand for each product
Bas	Array of the number of bins in a stack of each product
Bs	Calculated array of demand of stacks of each product (from D and Bas)
Sl	Array of storage locations of each buffer unit
MaxMBCLength	Integer of the max depth of an MBC cluster
MBCException	Array of exceptions on the Max MBC Depth
Geometry	Geometry of the environment
AvailableTime	Integer of the available time of an AGV in a shift
AGVTravelInfoHeaders	Text array of the info in the ATI array
ATI	Array of AGV travel inputs
IntroMBC	Array of number of actions of an AGV on the job: intro to MBC
MBCDSCAB	Array of number of actions of an AGV on the job: MBC to DSCAB
FinishMBC	Array of number of actions of an AGV on the job: Finished pproducts to MBC

Parameters we need to calculate are the amount of bin stacks per product $B_{s,i}$ and the AGV work time AT. In equation 5.1 the formula for $B_{s,i}$ is depicted. Important is to round up to the next integer since the model does not work with half stacks.

$$B_{s,i} = \lceil D_i / B_{as,i} \rceil \qquad \forall i \in I \tag{5.1}$$

AT can be calculated by

$$AT = \sum_{n=1}^{N} (MBCDSCAB_n + IntroMBC_n) * ATI_n * (\sum_{i=1}^{I} (Bs) - Bs_{-1}) + FinishMBC_n * ATI_n * Bs_{-1} \tag{5.2}$$

5.4.2. Indices and sets

Since our model should be able to evaluate a variety of different configurations, our sets are dependent on the length of our inputs. These sets are depicted below.

Index	Info	Set
i	Products	i in I with range length(D)
j	Storage units	j in J with range length(SI)
k	MBC locations behind MBC master	k in K with range (MaxMBClength)
l	Vertical storage locations of an MBC	l in L with range (SI[2])
m	Possible locations for MBC master	m in M with range length(D)
n	Different info points of AGV data	n in N with range length(ATI)
p	Grid locations not in wasted space	p in P with range length(MBCExceptions)

5.4.3. Decision variables

For our model to be able to find a solution, variables on which the solver may decide are introduced. Here the variables are depicted and explained.

$$X_{units,j} \tag{5.3}$$

In equation 5.3 the most important variable is introduced. X_{units} is an integer array that will hold the amount of each system unit J.

$$MBC_{grid,lmk} \tag{5.4}$$

To be able to track and visualize the amount of MBC masters and HBTs used, a grid of the available locations is created. MBC_{grid} is a binary, three dimensional matrix where a 0 indicated no storage location and a 1 indicates there is a storage location. The X-axis indicates the vertical amount of storage locations of an MBC. The Y-axis shows the horizontal amount of MBCs. The Z-axis shows us the depth of a MBC row (number of HBTs behind a master).

$$MBC_{usl,lm} \tag{5.5}$$

The front (XY plane or LM plane) of the matrix MBC_{grid} indicates the amount of USLs. The size of these storage locations is stored in MBC_{usl}. MBC_{usl} projects the amount of stacks that can be stored in an USL on the LM plane.

$$Assign_{lmi} \tag{5.6}$$

The binary matrix $Assign$ keeps track off if and where a product is stored. The front plane is the same as MBC_{usl} and the depth of this matrix equals the amount of different products. Each LM plane represent a product i, and a location $[l, m]$ equal 1 if product i stored there. For example: if product 4 is stored in $MBC_{usl}[2, 3]$, then $Assign[2, 3, 4]$ will equal 1.

$$C_{b,j} \tag{5.7}$$

Cost variable for the bill of material costs for system unit j

$$C_{a,j} \tag{5.8}$$

Cost variable for the assembly costs for system unit j

$$C_{i,j} \tag{5.9}$$

Cost variable for the installation costs for system unit j

$$C_{f,j} \tag{5.10}$$

Cost variable for the floorspace costs for system unit j

5.4.4. Objective function

The objective function of our model describes our goal of what we want to achieve. In our case a minimization of the total costs is required, where all costs weigh equally heavy. This results in the following objective function.

$$MINIMIZE \qquad C_{total} = \sum_{j=1}^{J}(C_{b,j} + C_{a,j} + C_{i,j} + (C_{f,j} * OD)) \tag{5.11}$$

5.4.5. Constraints

To describe the systems workings and limitations, constraints are formulated. In this section these constraints are depicted and elaborated as to why they are required. The constraints are divided in general MBC, dedicated storage, unit assignment, cost and non-negativity constraints.

General MBC constraints

$$MBC_{grid}[l,m,k] = MBC_{grid}[0,m,k] \qquad \forall l \in L, \forall m \in M, \forall k \in K \tag{5.12}$$

The MBC is always X locations high. This value X is given by Sl[2] and equals in this case 4. This is a mechanical design constraint of the MBC unit. Equation 5.12 says if an MBC is placed in position [m,k] all the l position in that location should also equal 1. If [m,k] does not contain an MBC, the l values in that column also contain no MBCs.

$$MBC_{grid}[l,m,k] \geq MBC_{grid}[l,m,(k+1)] \qquad \forall l \in L, \forall m \in M, \forall k \in (K-1) \tag{5.13}$$

Placement of MBCs should always start at the first row. Equation 5.13 says the value of the location in front of an other location should always be larger or equal. Since MBC_{grid} is binary, this means there can never be a 1 behind a 0.

$$MBC_{usl}[l,m] = \sum_{k=1}^{K} MBC_{grid}[l,m,k] \qquad \forall l \in L, \forall m \in M \tag{5.14}$$

MBC_{usl} should indicate the amount of stacks that fit in each unique stack location. To achieve this the values of the rows behind each [l,m] location in MBC_{grid} are added.

$$\sum_{k=1}^{K} MBC_{grid}[0,m,k] \leq MBCmax[m] \qquad \forall m \in M \tag{5.15}$$

To enable exceptions on the maximum MBC depth parameter, constraint 5.15 is introduced. $MBCmax$ is an array indicating the max depth of each individual row. This constraint checks to validity of row placement according to $MBCmax$.

Dedicated storage constraints

$$\sum_{i=1}^{I} Assign[l,m,i] \leq 1 \qquad \forall l \in L, \forall m \in M \tag{5.16}$$

One unique storage location can only hold one product. This means if MBC_{usl} indicates a storage location has room for four stacks, and only two are used, the remaining two can not be used for a different product.

$$\sum_{l=1}^{L}\sum_{m=1}^{M}(Assign[l,m,i] * MBC_{usl}[l,m]) \geq Bs[i] \qquad \forall i \in I \tag{5.17}$$

All products should be placed in the end of the model. Equation 5.17 ensures this. Since $Assign$ is binary, we can multiply each the LM plane with MBC_{usl} and all locations where product i is not placed will equal zero as well. Next a simple summation of all non zero values gives us the amount of storage space there is for each product. By stating this amount should be greater or equal to the demand of that product ($Bs[i]$) complete storage of all product is assured.

Unit count constraints

$$X_{units}[1] = X_{units\_init}[1] \tag{5.18}$$

As stated in section 5.2, the number of SPELs required depends on numerous factors. In this model the initial number of SPELs stated in the $Xunits\_init$ input array.

$$X_{units}[2] = \sum_{m=1}^{M} MBC_{grid}[0,m,0] \tag{5.19}$$

The front row of MBC_{grid} is always filled with only MBC masters. This is because the HBTs lack the capability of interacting with an AGV. To find the number of MBC masters the first row of binary variable MBC_{grid} can be added together.

$$X_{units}[3] = \sum_{m=1}^{M}\sum_{k=2}^{K} MBC_{grid}[0,m,k] \tag{5.20}$$

To determine the number of MBC units, any MK plane of the binary variable MBC_{grid} can be summed up. This is possible due to equation 5.12. Since the first MBC in a row is always an MBC master, the sum over set K start at row 2.

$$X_{units}[4] = X_{units\_init}[4] \tag{5.21}$$

The amount of MBC slaves is zero in the current state due to only the front of MBC_{grid} being accessible. This value is therefore imported from the $Xunits\_init$ input array.

$$X_{units}[5] = X_{units\_init}[5] \tag{5.22}$$

The amount of DSCABs used is fixed by the mechanical constraints as explained in section 5.2. This value is therefore imported from the $Xunits\_init$ input array.

$$X_{units}[6] = AGVreq \tag{5.23}$$

The demand for AGVs is directly coupled to input $Demand$ and AGV travel times and is calculated separately. The model can not optimize the need for AGVs since travel time is defined as constant. Even tough placement of the system has a direct influence on travel times, and thus on AGV requirement, this influence is ignored as stated in section 3.9

Cost constraints

$$C_{b,j} = X_{units}[j] * BOM[j] \forall j \in J \tag{5.24}$$

Bill of material equals number of units of type j times their BOM costs

$$C_{a,j} = X_{units}[j] * AssemblyTime[j] * FTE_c \forall j \in J \tag{5.25}$$

Assembly costs equal the time it takes to assemble a unit of type j, times the number of units of type j, times the cost of one FTE

$$C_{i,j} = X_{units}[j] * InstallationTime * FTE_c \forall j \in J \tag{5.26}$$

Installation costs equal the time it takes to install a unit of type j, times the number of units of type j, times the cost of one FTE

$$\begin{aligned} C_{f,j} = {} & OD * FloorspaceWriteoff * \\ & (Geo[1] + \sum_{p=1}^{P} \sum_{k=1}^{K} MBC_{grid}[0,p,k] * Footprint[2] + X_{units}[1] - 1 * Footprint[1]) \end{aligned} \tag{5.27}$$

Floorspace cost include wasted space, which is included in Geo, and units placed outside of the wasted space. Through set P the model can quantify the cost benefit of placing a unit in the wasted space or outside of it due to set P not spanning as for as set M.

Non-negativity constraints

Here the lasts constraints on the decision variables are stated. These constraints are meant to prevent the decision variables to become negative.

$$X_{units,j} \geq 0 \qquad \forall j \in J \tag{5.28}$$

$$MBC_{grid,lmk} \geq 0 \qquad \forall l \in L, \forall m \in M, \forall k \in K \tag{5.29}$$

$$MBC_{usl,lm} \geq 0 \qquad \forall l \in L, \forall m \in M \tag{5.30}$$

$$Assign_{lmi} \geq 0 \qquad \forall l \in L, \forall m \in M, \forall i \in I \tag{5.31}$$

$$C_{b,j} \geq 0 \qquad \forall j \in J \tag{5.32}$$

$$C_{a,j} \geq 0 \qquad \forall j \in J \tag{5.33}$$

$$C_{i,j} \geq 0 \qquad \forall j \in J \tag{5.34}$$

$$C_{f,j} \geq 0 \qquad \forall j \in J \tag{5.35}$$

5.5. Model assessment

As mentioned in section 3.6, before the model can be used it has to be verified and validated. This is done directly since outcomes must be reliable to compare them with the current method. Finally the problems found in this verification and validation be used to modify the model to increase reliability and validity.

5.5.1. Verification

The implementation of the model constructed in section 5.4 is done in Python. During coding, mistakes or logic errors can be made. To ensure the model used is free of these errors the following tests can be done:

1. External code check
2. Extreme cases simulations
3. (Intermediate) results checking

To find coding errors it is best to have another expert take a look at the code. While coding, one might not see their own mistakes as mistakes. This is especially true for assumptions on how the code works. An external view often is refreshing and focuses on areas the creator would already have accepted as 'good'.

Extreme cases are situations the model should never encounter. How the model would react on these inputs can be reasoned. The following extreme cases are tested. First the required storage space is not enough. The demand is increased in such a way that the model does not have the capability to accommodate all the products. This should conclude in an infeasible model. Secondly we can increase the amount of product types and have a demand of 1 stack per product. There should be no MBC HBTs present during this setup. Thirdly we can increase the price of an MBC HBT. This will result in the max amount of MBC masters the model can place and remaining products will be stored in an MBC HBT.

The intermediate results and steps of the solver can be evaluated where possible mistakes may lay. When checking results it is not only important to check the result itself but also the sizes of matrices of the results. When multiplying M x N matrix with an N x L matrix, the results should equal a M x L matrix. These check indicate whether calculations are done correctly and dimensions are correct. An MBC HBT can only be placed behind a master or another HBT. This means there can never be more than three times the number of HBTs compared to masters. Simple results are also to be checked. A good example of a simple result in our model is the amount of AGVs required. Since the calculation for this result is easily done by hand it can be quickly compared to the model's prediction. In table below the tests, predictions and results are summarized.

		Prediction	**Outcome**
External code check			
	Done by college	Small code clarification needed	No operational errors, run time is long
Extreme cases			
	Demand to great	Model infeasible	Model infeasible
	Demand one per product, lots of different products	No MBC HBTs	No MBC HBTs
	Price of MBC HBTs very high	Max MBC masters	Max MBC masters
(Intermediate) results			
	Size MBC$_{grid}(nonzeros\&zeros)$	4x7x4	4x7x4
	$X_{units}[3] \leq X_{units}[2] * 3$	Correct	Correct
	Number of AGVs	1	1
	Result	Under analytical solution	95 % of analytical solution

Table 5.4: Summary of verification methods ant their outcomes

5.5.2. Validation

the next question that should be answered is: is our model the right model? In other words, did we describe the right real world problem. In order to answer this question the results of the model can be compared to known or analytically predicted results. Not only the final solution is of interest here. Analysing decision variables of the system may give more insight in how the model makes decisions. By varying the inputs for the system more knowledge on how the model reacts can be gathered.

First we check the current results of the MBC placement. To visualize this a grid is plotted in which the cabinets are placed. This can be seen in figure 5.2. The YZ-plane on X equals zero is the front view. From this plot it can be clearly seen here are two rows four, two of three, one of two and one of one MBC. The placement of in the Y direction does not have influence on the total costs so this is random. In table 5.1 is has been calculated there are at least 24 USLs are needed. The grid generated by the model meets this requirement.

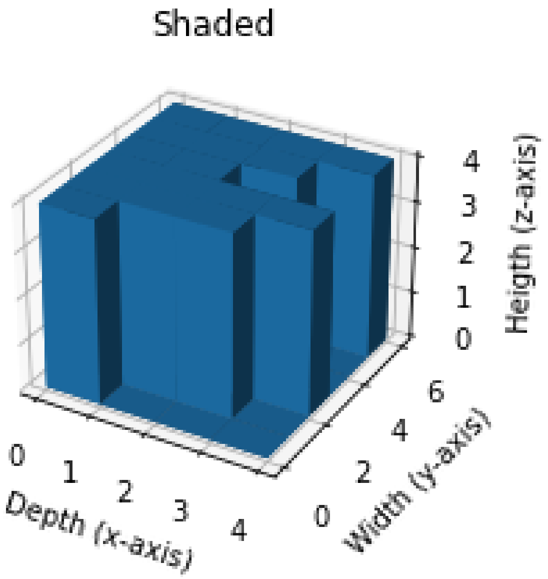
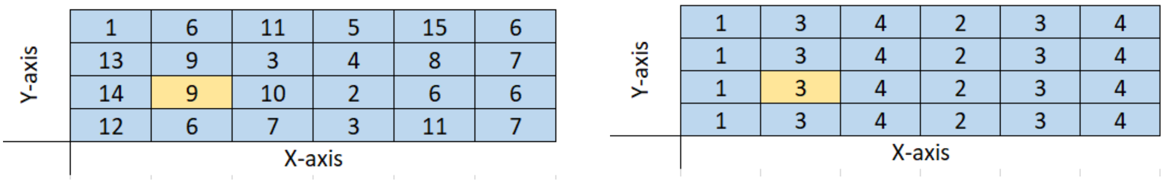


Figure 5.2: Plot of the MBC cluster as composed by the model

The important thing is that there are no unnecessary MBC HBTs placed. By calculating the space reserved for each product this can be checked. In figure 5.3 two front views of the MC cluster are shown. In subfigure 5.4a the product IDs are shown. This indicates where in the cluster these products are stored. In subfigure 5.4b the size of the USLs are shown in a front view of the MBC cluster shown in figure 5.2. Together with these two figures the excess of storage space can be calculated. This is shown in table 5.5. Here it can bee seen all requirements are met, and one stack location in the whole cluster is unused. This location is marked in yellow in figure 5.3. This empty location does not result in extra costs since the other locations of the MBC are required. Thus this results is as expected.

Product ID	**Stack demand**	**Allocated stack space**	**Quadruple stack demand**	**Allocated stack space quadruple demand**
1	1	1	4	4
2	2	2	8	8
3	6	6	24	24
4	2	2	8	8
5	2	2	8	8
6	17	17	68	68
7	12	12	48	48
8	3	3	12	12
9	5	6	20	20
10	4	4	16	16
11	7	7	28	28
12	1	1	4	4
13	1	1	4	4
14	1	1	4	4
15	3	3	12	12

Table 5.5: Comparison of required and allocated storage space


Y-axis
X-axis
Y-axis
X-axis

1	6	11	5	15	6
13	9	3	4	8	7
14	9	10	2	6	6
12	6	7	3	11	7

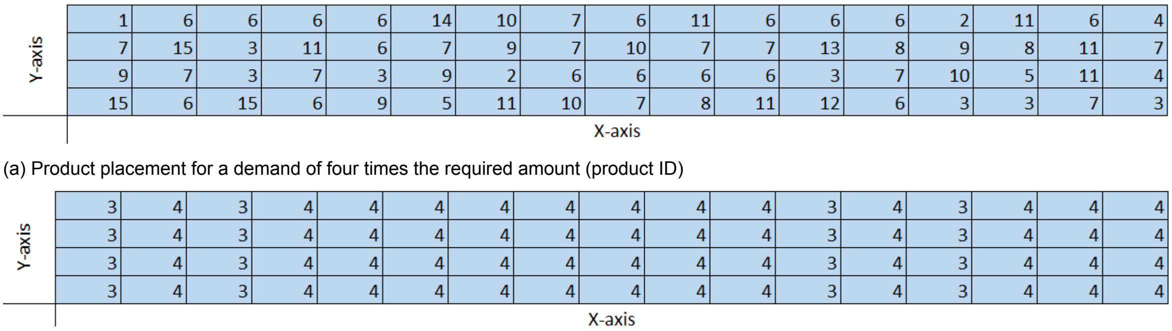
(a) Product placement in MBC cluster (product IDs)

1	3	4	2	3	4
1	3	4	2	3	4
1	3	4	2	3	4
1	3	4	2	3	4

(b) USL size projected on the MBC cluster (in number of stacks)

Figure 5.3: Front views of the MBC cluster

The next tests are varying the inputs and observing how the decision variables change. First the demand is varied and the empty stack locations can be calculated as done above. The demand is scaled up to four times the demand of each product. The important thing is to look for excessive MBCs. An MBC comes with four stack locations, thus a maximum of three unused stack locations may occur before eliminating the MBC is more effective. Since the model now has more inputs to decide on, empty spaces are most likely to be completely eliminated. As seen in section 4.2, the cost of an MBC master is double that of an MBC HBT. This means it will be more effective to place a HBT than to add a new collum. An MBC with a depth of one, such as the first column of figure 5.4b, is expected to become 4 MBCs deep. In figure 5.4 the front view the generated MBC cluster is depicted. Together with the results shown in table 5.5 we see the behaviour we expected from our model. There are no columns of one or two MBCs deep and no empty stack locations.


Y-axis
X-axis
Y-axis
X-axis

1	6	6	6	6	14	10	7	6	11	6	6	6	2	11	6	4
7	15	3	11	6	7	9	7	10	7	7	13	8	9	8	11	7
9	7	3	7	3	9	2	6	6	6	6	3	7	10	5	11	4
15	6	15	6	9	5	11	10	7	8	11	12	6	3	3	7	3

(a) Product placement for a demand of four times the required amount (product ID)

3	4	3	4	4	4	4	4	4	4	4	3	4	3	4	4	4
3	4	3	4	4	4	4	4	4	4	4	3	4	3	4	4	4
3	4	3	4	4	4	4	4	4	4	4	3	4	3	4	4	4
3	4	3	4	4	4	4	4	4	4	4	3	4	3	4	4	4

(b) USL size projected on the MBC cluster for quadruple demand (in number of stacks)

Figure 5.4: Front views of the MBC cluster for quadruple demand

Next the amount of AGVs needed are evaluated. From real world testing we know the current state can do with one AGV. For quadruple the demand this is not true. Since an usage of 80% is calculated for a single AGV in the current state, a four times as big a demand should result in at least three AGVs. Our model indicates two AGVs suffice with an usage of 87% both. This indicates the accuracy of the amount of AGVs is not good enough. In section 5.5.3 modifications to better estimate the amount of required AGVs are formulated.

5.5.3. Modifications

From previous sections a couple of problems are identified. In this section modification of the implementation and the model are suggested and implemented.

The first problem is the run time. The model takes 3808 seconds to solve for the demand indicated in table 5.1. This is very long, even for a relatively small amount and quantity of products. When calculating for larger sets, the run time is expected to grow even more. Two main factors for the long run time have been identified: the amount of loops and the size of set M. The code has been separated into many loops for cosmetic and clarification purposes. This results in the need to run through each loop in every iteration. By gathering all the constraints in the same loop, the run time is reduced to 512 seconds which is a improvement of 87 %.

The second factor, the size of set M, is more difficult to fix. The model should have the freedom to place as many MBC master as it wants, the size of M should not be a limiting factor. Since our model is quantitative, the downside of making M excessively large, is the model trying every option. When analytically it is easy to understand it does not matter where in the MBC_{grid} the zeros row will be placed, the model does not approach it this way. Every option will be tried, calculated and discarded if it not improves the objective function. From initial calculations it can be found the width of non-zero entries of M is 6. When setting the width of M to 6, instead of the number of products i, the run time decreases to 15 seconds, and yields the same result. This is an improvement of 99.6 %. However, the set M can not become a restriction on the choices of the model. While analytically it is always regarded as a more costly decision to start a new column, it might be necessary or counter intuitive to do so. In order to give the model the option to determine the length of M without increasing the run time heavily the code is adjusted. The size of M is increased by one until the model is feasible and a solution is found. Then $M + 1$ is also calculated and the solution is compared to the solution of M. When M has the best solution, the loop breaks. If the the solution of $M + 1$ is optimal, the same process is done for $M + 2$ and so on. Combining the methods above fastest run time of under 1 second. These modifications and their resulting improvements are summarized in table 5.6.

Modification	**Runtime (seconds)**	**Improvement**
Nothing	3808	-
Loop bundling	512	87%
Incremental increase of M	15	99.6%
Loop bundling & Incremental increase of M	<1	~100 %

Table 5.6: Run time improvement modifications

5.6. Results - total cost optimal solution

In this section the results of the current state model are discussed and compared to the current unit-assignment method results discussed in section 5.3. First the costs and system unit amount will be showed. Next the critical demand of this configuration is determined. Lastly the buffer time required by the system is varied. Due to confidentiality reasons, these results are shown as a percentage of each other. At each result comparison it is noted how they relate to each other. Results are based on the demand of the production environment at a takt time of 60 seconds as shown in figure 5.1.

5.6.1. Unit quantity and costs

Due to the constraints of the current implementation of the system, the model can not decide on the number of SPELS and DSCABs. These values are directly copied from the unit initialization input parameters. With this in mind, the required units are depicted in table 5.7. Here we can see the only difference is the number of required MBC HBT cabinets. This is as expected since other values ae not to be changed by the quantitative model due tot he system constraints.

	SPEL	**MBCPELM**	**MBCHBT**	**MBCPELS**	**DSCAB**	**AGV**
Quantitative model	4	7	14	0	16	1
Current assignment method	4	7	21	0	16	1

Table 5.7: Unit amount required by the current state model

This decrease in MBC HBT cabinets required will also be the driving factor behind the lower total costs. In table 5.8 the costs as determined by the model are compared to the analytical solution. The analytical solution is taken as 100%. In figure 5.5 the different costs from the model optimal solution are depicted and how much they contribute to the total costs.

	Total	BOM	Assembly	Installation	Floorspace	Floorspace (m2)
Analytical solution	100%	100%	100%	100%	100%	114,1
Model optimization	94%	94%	87%	87%	97%	110,2

Table 5.8: Costs as calculated by the quantitative model in euros

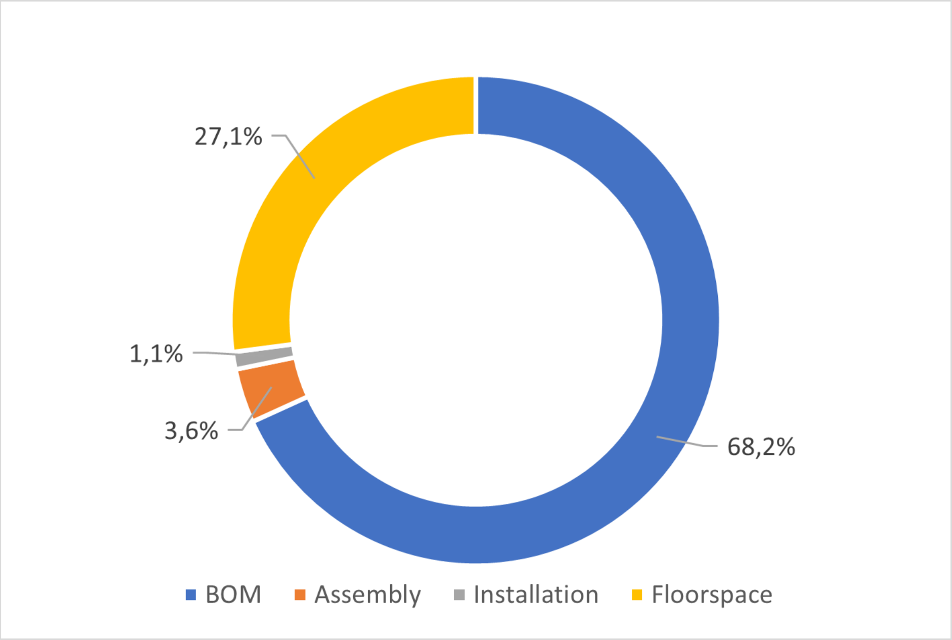


Figure 5.5: Buildup of different costs from total costs

5.6.2. Critical demand

Takt time for one product is currently 60 seconds. To scale up production one has multiple options. One being shortening the takt time. Improvement on the production process will eventually lead to this shorter takt time and thus to a higher output of the production line. This higher output also required a higher demand of parts for the individual cells. The question 'how much more can the demand grow before new buffer units are required' is therefore important. By incrementally increasing the demand by 0.5% in our model and checking the resulting required units, the critical demand of this solution can be checked. These results are depicted below in table 5.9.

Critical demand analytical solution	102 %
Deciding factor analytical solution	Number of unique storage locations
Critical demand model optimal solution	102 %
Deciding factor model optimal solution	Number of unique storage locations

Table 5.9: Critical demand

The critical demand for the current storage assignment method as described in section 5.3 can also be determined. Since in this method an USL always has 4 stack locations, the deciding factor whether or not the costs increase is the number of USL required, not their size. This is different from our quantitative model. There the USLs are not always as deep as possible, meaning an increase in demand can first lead to adding a MBC HBT rather than adding a new MBC master to increase the number of USLs. Through this comparison it can be concluded the critical demand of the current assignment method is 102 % as well, even tough the costs of implementation are higher.

5.6.3. Takt time modification

Buffering is broad term as described in section 2.3. How long production can continue without the need for human operators or restocking depends on the amount of products stored in the system. The more products need to be stored, the more units are required. In this section the demand is modified by modifying the takt time of a product. These results are shown in figure 5.6. Results are shown as a percentage of their respective value at a takt time of 60 seconds (base value for 100%). Notable are the high costs at very low demand. At a takt time of 300 seconds (20% demand modifier) total costs only have decreased by 15 %. This can be explained by the fact that even at low demand, all the DSCABs are still needed in the current system design. Also, due to the dedicated storage strategy, the same amount of USLs are required since different products can not be stored in the same USL. Higher takt times also do not increase total system costs with the same proportions. As can be seen in the figure, a takt time of 30 seconds (200% demand modifier) will only result in a 20% growth in costs.

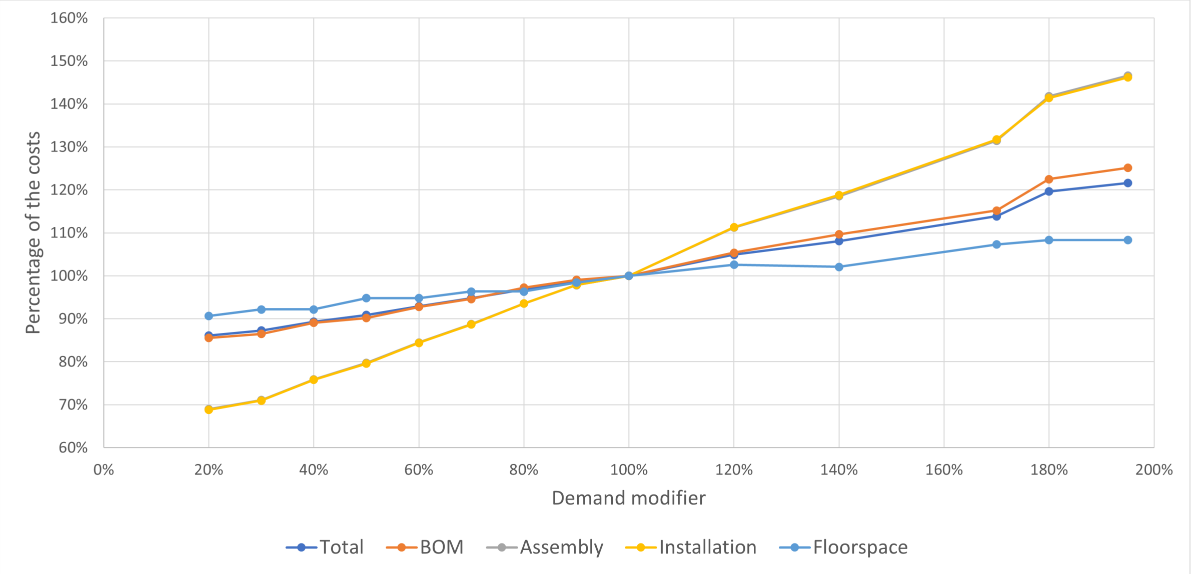
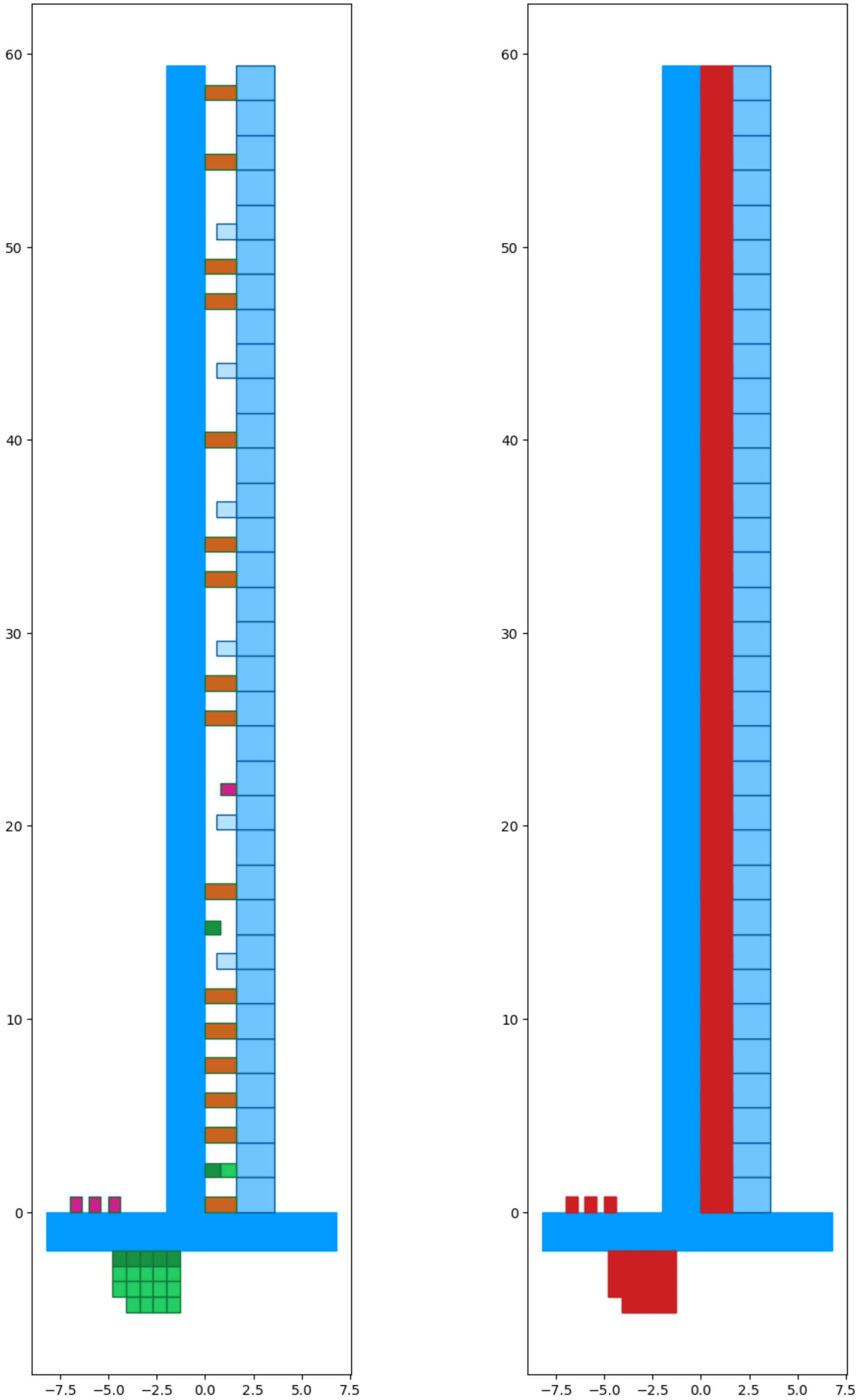


Figure 5.6: Costs in relation to the demand, as a percentage of that cost source at 100% demand

5.6.4. Generative design

Our model also generates a floorplan based on the optimal solution found. In figure 5.7 the generated solution is depicted. Here we can see placement of MBC units only occurs outside of the wasted space area. This indicates it is more cost effective to be able to expand rows to 4 MBCs then to save space.



(a) Environment

(b) System footprint

Figure 5.7: Model optimal solution to system implementation

5.7. Results - BOM and floorspace optimal solutions

Our model also includes the function to only optimize for BOM or floorspace. These results are shown in table 5.10. Results are shown as a percentage of the results from the total cost optimal solution from section 5.6. In figure 5.8 the environments generated by the model are depicted.

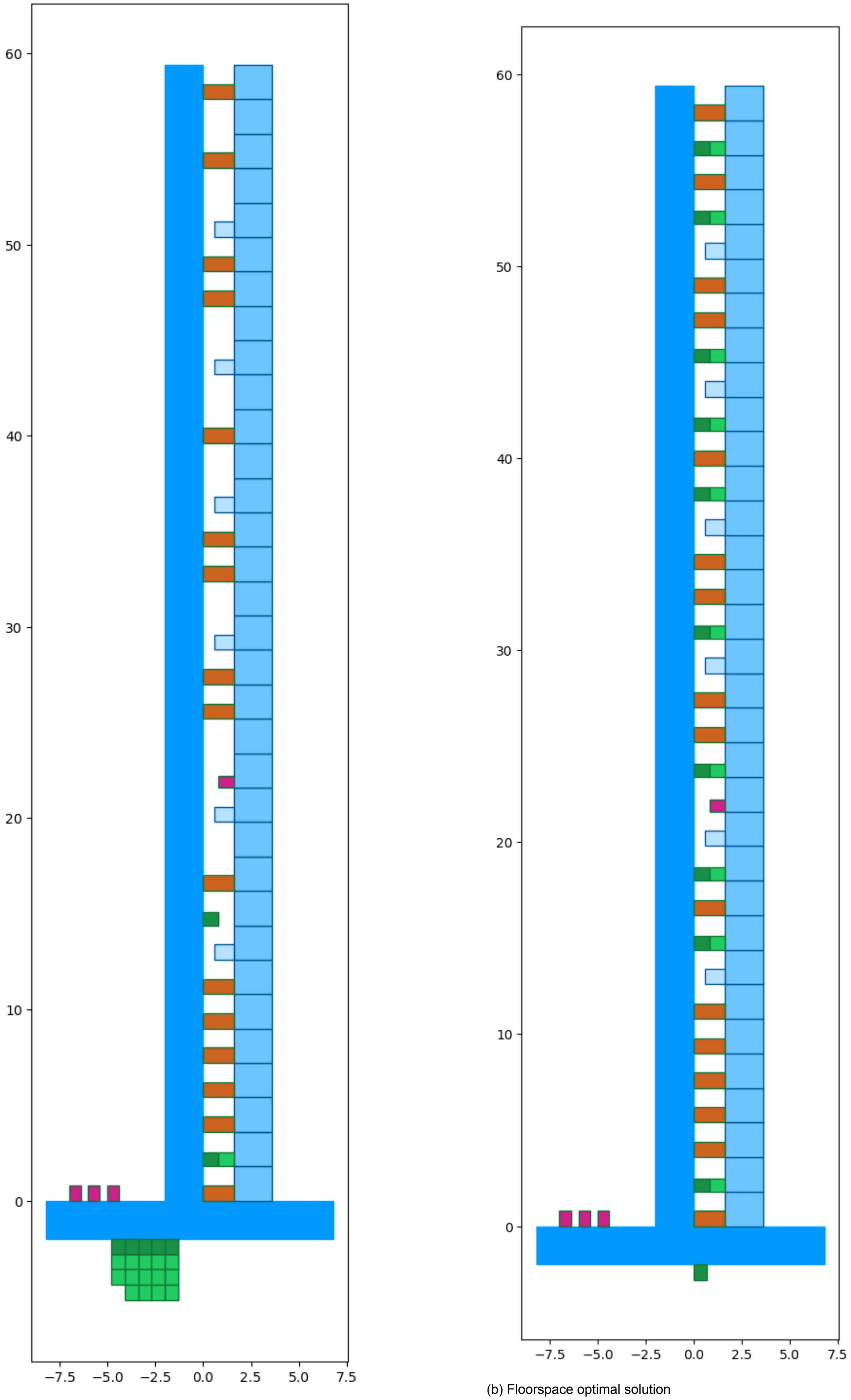
	Total	BOM	Assembly	Installation	Floorspace	Floorspace (m2)
Total	100%	100%	100%	100%	100%	107.52
BOM	100%	100%	100%	100%	100%	107.52
Floorspace	101%	105%	104%	105%	92%	98.56

Table 5.10: Result comparison of different optimal solutions

5.7.1. Floorspace depreciation cost alteration

From results show above, it can be seen how the total cost and BOM optimal solution are the same. This indicated the BOM costs grow more rapidly than the floorspace costs. An interesting thing to see is if our model can find a balance between both cost factors. This can be done by increasing the depreciation cost per square meters in the production hall. This is helpful to check since values are based on actual depreciation costs which are specific to Prodrive Technologies. Different situations can have higher, or lower depreciation costs and our model should be able to perform in these cases as well. In this test the depreciation costs are increased by 20%.

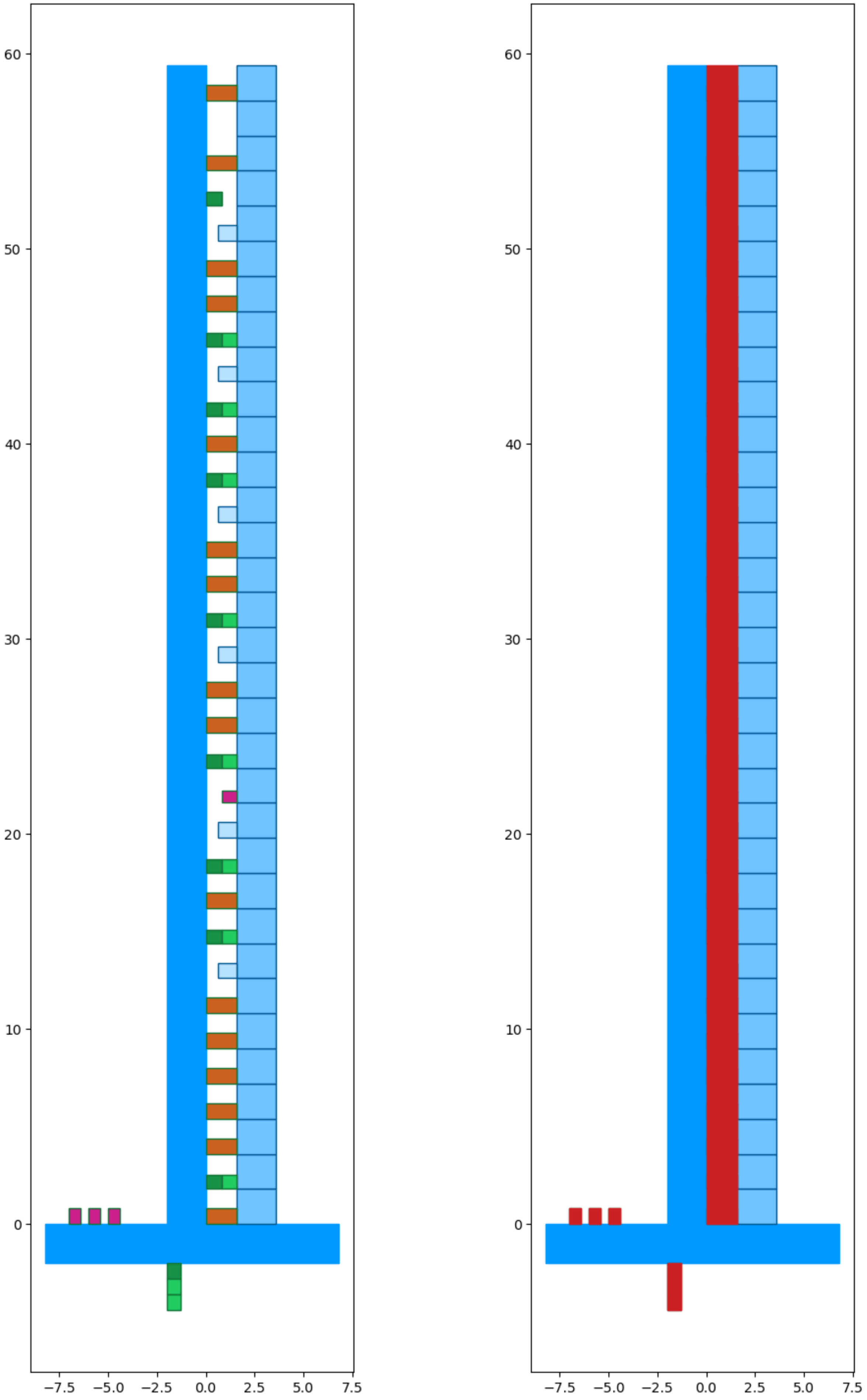
In figure 5.9 the resulting environment is depicted. Here the balance between choosing a longer row outside of the wasted space area, and placing more short rows inside of it can be seen. The total system footprint in this setup is 99.12 m^2, which is just a little higher than the footprint optimal solution.



(a) BOM optimal solution

(b) Floorspace optimal solution

Figure 5.8: Different optimal solutions



(a) Environment

(b) Footprint

Figure 5.9: Total cost optimal solution with an 20% increase in depreciation costs

5.8. Chapter summary

This chapter aims to answer the subquestion ”How does a quantitative cost optimal solution of a bin handling system relate to an analytical solution?”. Analytical solutions are often based on safety since a ’just enough’ solution is hard to determine. Interesting is the identical critical demand depicted in section 5.6.2. Here it can be seen the analytical can not handle an increase in demand better than the model optimal solution. This is due to the lack of USLs and the dedicated storage constraint. The analytical solution contains more empty stack locations within the USLs since all USLs are considered to have 4 stack locations. This unnecessarily increases the BOM costs. Often through usage these over-capacities are noticed and capacity is altered. However, this means the units still are bought.

The analytical solution found in the current implementation method in the case study did in first not include a clear floorspace cost factor. To accurately inspect the implementation costs of any system, one should have a clear definition on how floorspace costs are handled. By clearly quantifying the floorspace costs, better informed decisions can be made on implementation. As seen in section 5.7.1, the quantitative model is able to find a optimal balance in floorspace and BOM costs. Such equilibriums are tedious to find analytically. For quantitative models, more work needs to be done to set up the model. Once a model is constructed, analyzing new configurations, demands and units properties becomes quicker and more reliable. This way predictions can be made on how different costs will increase.

III

Design strategies

6 Scenarios

The current state model is bound by the mechanical limitations of the current system. However, design of bin handling systems can vary. In this chapter a start is made in answering the subquestion: *'What is the impact of different design strategies?'* This is done through real world, as well as purely theoretical experimental setups. In chapter 7 the experimental setups are simulated and their results are shown in order to fully answer the subquestion

6.1. Experimental setup boundaries

To be able to compare different experimental setups, most parameters should be kept constant. These parameters are identified and explained in this section. The first constant factor is the production environment. Placement of production cells is fixed. Experimental setups cannot assume altered placement of production cells to improve their performance. This would introduce to much variables to reliably compare different experimental setups. The same goes for the bins in the bin handling system. The focus of this thesis is hardware of bin handling systems, and not the bin design. This is why bin type, shape and size are kept constant in each experimental setup. (semi) Product design can not be altered in order to improve throughput. One might opt to redesign products in order to fit the bins better and thus increasing the products per bin. While this is a sound optimization strategy for very large volumes, it is not considered an option in these experimental setups. Takt time, and thus demand, is kept at 60 seconds. This is the same as in the case study from chapter 4.1. To test the experimental setups on their ability to cope with demand increase, this takt time of 60 is used as a base value to scale the demand.

All other factors that may be of influence are based on the case study data. If not specifically noted, they remain unchanged. When factors are altered, the difference in value is estimated and reasoned why this value is chosen. In table 6.1 the different parameters are depicted once more.

Experimental setup parameter alteration

Fixed	Can be altered individualy
Production cell placement	Unit design
(semi) Product design	Unit placement
Bin type, shape and size	BOM
Takt time / demand	Assembly time
Operating days	Installation time
AGV docking time	Footprint
AGV leaving time	AGV travel time
AGV (un)loading time	

Table 6.1: Parameters in experimental setups

To facilitate some experimental setups, assumptions on supporting systems have to be made.

Mostly a high level of autonomy and planning of the external support LES system. As stated in section 1.5, the LES system is treated as an external system. This means alterations needed in the LES system are not within the scope of this research. A note will be made of changed requirements by the bin handling system in each experimental setup. Impact of the following changes in the LES system are not included in the results. However, some calculations on the feasibility of experimental setups have to be done. Mainly timing wise. These calculations are done to prove experimental setups are feasible but are not further investigated for impact on other systems. An other assumption made in experimental setups is the bin order of a stack. When adopting a strategy where different cells are served from one stack, the order of the bins in the stack is assumed to be correct. Since the bin handling systems in the experimental setups have a clear introduction point, the stacks which are introduced should be in the right order. How this is accomplished, whether being with an FTE or order picking machine, is outside of this scope. Some experimental setups bring with them new restrictions on unit placement or unit interaction. These restrictions or introduced constraints are further reasoned in each relevant section. Further possible assumptions needed for experimental setups are indicated at the experimental setup's description.

6.2. Storage capabilities of the DSCABs

The (de)stacker units are equipped with two DSMs as can be seen in figure 4.4. The workings of these DSMs make it that they can hold one stack of bins. The holding of this stack is not technically seen as storage. However, holding an entire stack could be enough to run one shift. When considering production cell 13 from table 5.1, this can be seen. This production cell only needs 4 bins per shift. Since a stack has 7 bins, this cell could produce for almost two shifts with one stack of bins. Holding this stack of bins in the DSCAB rather then reserving an USL for this one stack eliminates the need for one USL with a size of one. Also, the AGV does not have to travel from the introduction SPEL to the MBC. It can now travel from the introduction station directly to the production cell. This eliminates one job for the AGV. The empty bins can also be stored in the DSCAB. As can be seen in figure 4.7, there is a second DSM. This DSM gathers the empty boxes. These empty boxes can stay in the DSCAB for one shift since there are only four empty boxes at the most.

6.2.1. Enhanced storage capabilities of the DSCABs

Current design of the DSCABs allow for storage of one full stack of bins. Now the question arises: what will happen if the design of the DSCABs allow for more storage. In this experimental setup the DSMs of the DSCABs have additional storage locations. Namely an addition 2 stack locations in each DSM, making the total 3 stack locations. With the addition of these storage locations the costs of a DSCAB is expected to increase. This increase is estimated to be 30%.

6.3. Shared storage space

Current units in the case study are not equipped with scanners. This means they can not keep track of product movements accurately enough to mix products. This results in the need for dedicated storage. With dedicated storage an USL can be assigned a single product ID resulting in easier product identification. Since an MBC HBT costs half of an MBC master, creating new USLs is more costly then expanding existing USLs. In this experimental setup, the assumption is made that shared storage strategies are allowed.

6.3.1. No MBC row limit

Due to restrictions related to the available space on the production floor, the case study has a maximum amount of MBC units that can be placed behind each other. This is most likely the case for more production environments. The question now arises: what if there is a limit of 10, or no limit at all? Since it is known that expanding the row of MBCs is cheaper than adding a new one, different optimal solutions might be reached. In this experimental setup a shared storage strategy is also applied.

6.4. Central DSCAB

Since the stacking - destacking action is crucial to enable the transport of bins in stacks instead of single bins, a (de)stacker is required in bin handling systems. In the system described in the case study, such a (de)stacker is placed in front of each individual production cell. In the experimental setup discussed in section 6.5 the possibilities of having mobile DSCABs are discussed. In this experimental setup the stacking and destacking for the entire production line is done in a central location. AGVs bring single bins to and from the different cells. In order to make this experimental setup viable, a couple changes to the deployment of the units is made. Loading/unloading from the back of the MBC cluster is required to ensure the AGVs to not collide with each other when transporting to the cells and the DSCAB. This means MBC slaves are needed on in the back of the cluster, an extra AGV is needed on the back of the of the cluster to move stacks between the MBCs and DSCABs. Behind the DSCAB a SPEL is needed for loading and unloading on the back since here no docking module is present in the current DSCAB. In figure 6.1 this needed configuration is depicted. Note the MBC cluster is not optimized, this is for visualization purposes only. One more DSCAB is required at the end of the production line at cell F since the bins with finished products should not be mixed with the empty bins coming back from the other cells. This experimental setup required a high level of planning from the logistic execution system since bins are mixed. This is assumed to be available. From figure 6.1 it can also be concluded the floorspace required also increases.

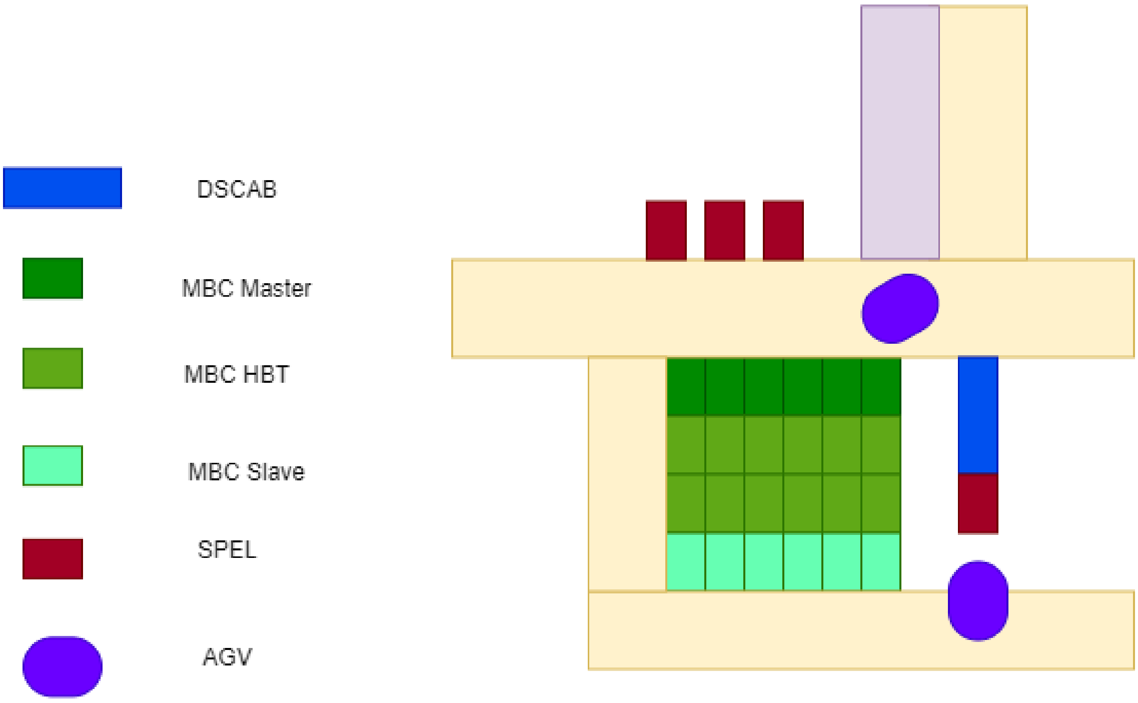


Figure 6.1: Centralized DSCAB experimental setup unit requirement example

6.5. Mobile DSCABs

As mentioned in section 4.1, the DSCABs are needed to destack the bins and feed single bins in to the production cells. In the case study, this is done through placing a DSCAB in front of each cell. As seen in section 4.2 the DSCABs are the most costly units of the system, both in BOM and floorspace. For some cells the demand per shift is so low the DSCAB is idle for most of the shift. In this experimental setup the DSCAB is modified to enable movement of the unit. This movement is a basic one directional movement along the length of the production line. This movement can be realized by rails or wheels in order to keep costs low. The BOM costs for such a unit are estimated to be 100 % higher than the original DSCAB discussed in the case study.

6.5.1. No obstacles

Since the movement is only one directional, the modified DSCAB can not pass an obstacle such as a liquid reservoir. The same experimental setup can be run when assuming there are no obstacles is the path of the DSCAB. Then the question becomes: what is the next limiting factor? In this experimental setup the crab like movements of a DSCAB could be extended over the entire length of the production line.

6.6. AGV with (de)stacker function

Combining functions of units could lead to better handling and reduced costs. In this experimental setup the possibilities of having a stacker - destacker unit in the AGVs is simulated. In order to enable this experimental setup, some assumptions have to be made. The first being the increased BOM costs of the AGVs. This is estimated to increase by 50% when outfitting the AGV with two stacker - destacker modules. Next is limited interactions with other units such as the MBCs. With the two (de)stacker modules, the AGVs will be high and more bulky. AGV travel speed is therefore reduced to 0.7 m/s.

6.7. Chapter summary

In this chapter experimental setups are introduced in order to evaluate different system configurations. This with the goal to set up the frame work to find find the answer to the subquestion: **What is the impact of different design strategies?**. In order to compare and depict experimental setups more easily, the experimental setups are given an ID. These IDs are depicted below in table 6.2

Experimental setup	Experimental setup ID
Case study	1
DSCAB storage included	2
DSCAB enhanced storage included	3
Shared storage	4
Shared - no MBC row limit	5
Central DSCAB	6
Mobile DSCAB	7
Mobile DSCAB no obstacles	8
AGV with (de)stacker function	9

Table 6.2: Experimental setup IDs

7 Results

In chapter 6 the foundation is laid to answer the subquestion: *What is the impact of different design strategies?* In this chapter the experimental setups are simulated and their results are shown per experimental setup, next the results are compared to each other. Generated designs for the production environment are gathered in appendix B.

Results are always displayed relative to the optimal solution found in the case study in chapter 5, unless stated otherwise. For example, the assembly costs source of the BOM optimal solution of experimental setup 2 is depicted as a percentage of the assembly costs source of the BOM optimal solution of the case study (setup 1).

7.1. Storage capabilities of the DSCABs

When including the storage locations in current DSCABs storage demand for the MBC cluster reduces by one stack for each product. This is shown in appendix C table C.1. Usage of this storage location in the DSCABs does not increase the costs since this USL is already there. The main modification needed is planning of the product flow, which is handled by LES. Results of the optimal solutions are depicted in table 7.1 with respect to the case study.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	96%	96%	91%	91%	98%	107,95
Floorspace	96%	95%	91%	91%	99%	98,43

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**
BOM	4	6	11	0	16	1
Floorspace	4	9	8	0	16	1

Table 7.1: Results of the DSCAB storage experimental setup w.r.t. the case study cost source optimal solution

In table 7.1 it can be seen how both optimal solution save 4% on the total system costs. The 4 % decrease in total costs is notable since this system does not need any investments in order to operate. This decrease can be explained by the decrease in number of MBCs required. There is a difference of almost 10 m^2 between both optimal solutions, even though there difference in total costs is comparable. This indicated the savings in floorspace are comparable to the increase in BOM costs due to these savings.

7.1.1. Enhanced storage capabilities of the DSCABs

When increasing the storage capacities of the DSCABs, even less storage space in the MBCs is required. The BOM of the DSCABs however increases. In table 7.2 the results of this experimental setup are depicted. Where experimental setup 2 lowered total costs, experimental setup 3 increase total costs by 5-6 %. The BOM costs increase even though the total number of units required by the system decreases. This is due to the higher costs of the DSCABs after modification. System footprint for this experimental setup's floorspace optimal solution is equal to the footprint of experimental setup 2. Since there are 20 locations in the wasted space area, which is already included in the footprint, all MBCs fit in this area. Placing less MBCs thus does not decrease the footprint further.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	105%	111%	76%	81%	96%	105,71
Floorspace	106%	110%	76%	81%	99%	98,43

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**
BOM	4	4	9	0	16	1
Floorspace	4	7	6	0	16	1

Table 7.2: Results of the enhanced DSCAB storage experimental setup w.r.t. the case study

7.2. Shared storage space

The shared storage space strategy is where one USL can be used for multiple different products at once. From the results in table 7.3 it can be concluded this strategy has almost no impact. The floorspace optimal solution is exactly the same as the floorspace optimal solution of the case study. The BOM optimal solution saves just 1% of the total costs. This 1% is due to the capability to extend an MBC row once more instead of adding a new row. This results in the one less MBC master and one more MBC HBT. This is as expected since we have seen in table 5.5 that the case study optimal solution only has one excessive stack location.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	99%	99%	100%	99%	100%	110,19
Floorspace	100%	100%	100%	100%	100%	98,99

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**
BOM	4	6	15	0	16	1
Floorspace	4	11	10	0	16	1

Table 7.3: Results of the shared storage strategy experimental setup w.r.t. the case study

However, shared storage space strategies might be help full for lower volumes of more variety of products. In order to check this, the model is used again with a much lower demand, only 20%. These results are shown in table 7.4. Here it can be seen the shared storage strategy is more use full.

Experimental setup	Total	BOM	Assembly	Installation	Floorspace
1	100%	100%	100%	100%	100%
4	97%	96%	95%	95%	99%

Table 7.4: Optimal solution experimental setup 3 with a takt time of 300 seconds w.r.t. the case study

7.2.1. No MBC row limit

Since there is no row limit and different products can be stored in the same USL, the model will opt for one long row in the BOM optimal solution. This can be seen in table 7.5. As this row only has one access point for the AGVs, one more is added on the back of the row through the addition of an MBC slave. The main reason the BOM costs are lower in this experimental setup is due to the difference in BOM cost between an MBC master and an MBC HBT unit. Since the floorspace optimal solution will opt for placement of units in the wasted space area, and the max row limit here is only 2, this optimal solution is identical to the case study optimal solution.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	94%	92%	89%	89%	98%	107,95
Floorspace	100%	100%	100%	100%	100%	98,99

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**
BOM	4	1	15	1	16	1
Floorspace	4	11	10	0	16	1

Table 7.5: Results of the no MBC row limit experimental setup w.r.t. the case study

7.3. Central DSCAB

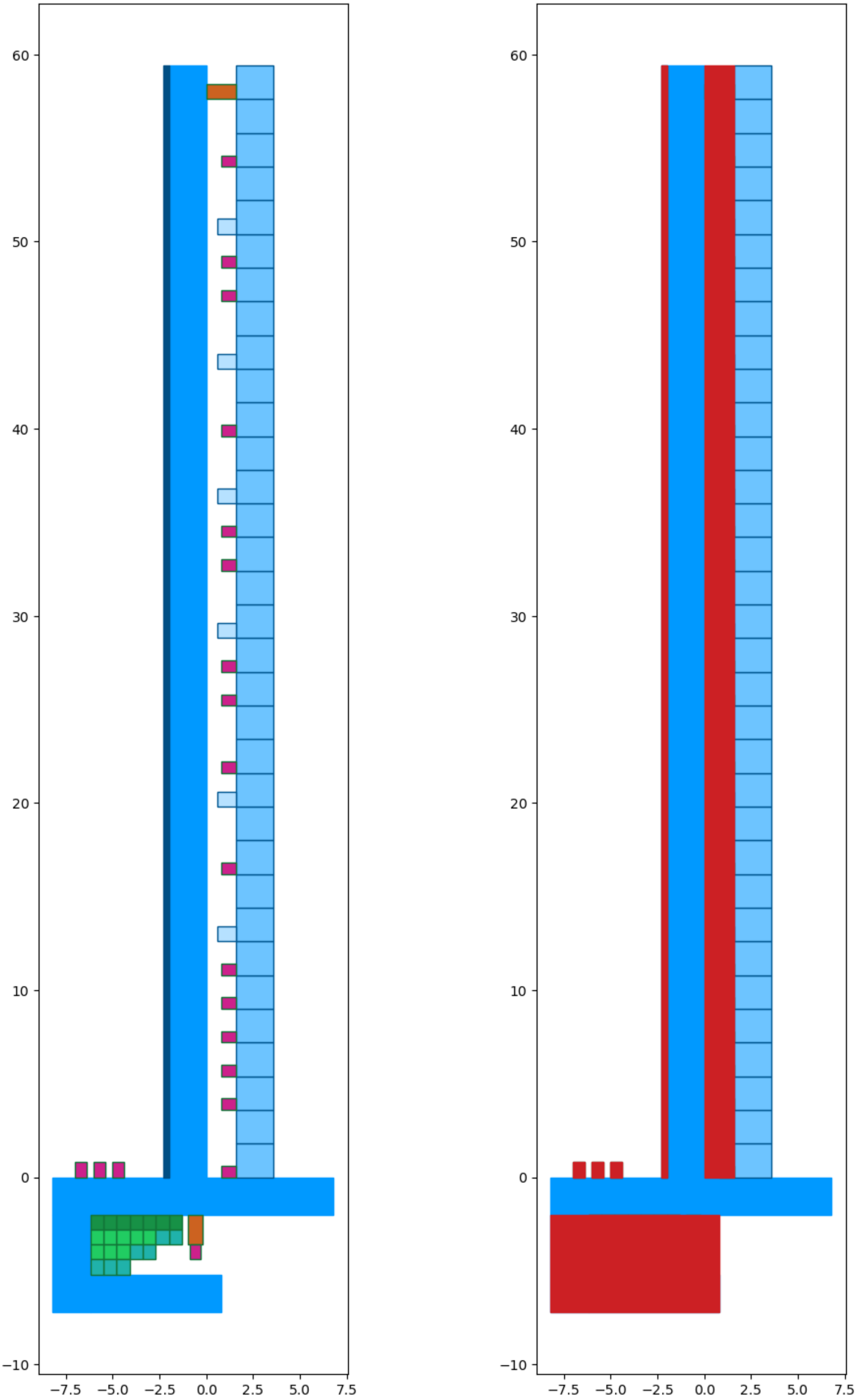
When implementing the experimental setup of one single, central DSCAB, the number of DSCABs required reduces drastically. This happens at the cost of requiring more of the other units. In table 7.6 the results of this experimental setup are depicted. From table 7.6 the costs seem to approve by about 15 %. While promising, there are downsides in this experimental setup. Since a production cell can only hold one bin, replacing the empty bin with a full bin could slow down the process. Replacing the empty now takes more actions: approaching, (un)loading and departing, twice. This has to be done twice since one AGV takes the empty bin and another AGV brings the full bin. In table 4.1 the duration of each of these actions can be found. This totals to 20 seconds for each AGV, resulting in 40 seconds required for restocking the supply at a cell, leaving only 20 seconds for the use internal transport and product picking. While this is enough, it strictly limits the takt time to a theoretical 40 seconds, not including the time it takes for internal transport and product picking.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	85%	69%	109%	99%	122%	134,82
Floorspace	85%	67%	108%	97%	136%	134,82

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**
BOM	19	7	8	7	1	6
Floorspace	19	7	8	7	1	6

Table 7.6: Results of the single DSCAB experimental setup w.r.t. the case study

During this experimental setup some reasoning is needed on the workings of the system and the introduced constraints. First there is the problem of the docking capabilities of an AGV onto the production cells. The cells do not contain a docking module. Therefore, each cell needs to be outfitted with a SPEL in order to be able to receive bins from the AGV. BOM costs of a SPEL are only 13% that of an DSCAB, so total BOM will still be lower. Since they are placed in the wasted space area, they will also not contribute to the total system footprint. Since there are now multiple AGVs operating at the same time there might be not enough room in the walkay for the AGVs to pass each other. An AGV has a width of 0.9 meter, so they should be able to pass each other. The length of an AGV is 1.1 meter. This means when a AGV is unloading, and it is turned 90 degrees, the walkway which is 2 meters wide will be to narrow. This is because in the (un)docking process, an AGV moves a bit backwards. The walkway is therefore widened. This extra space is counted towards to total system footprint as well. In order to supply the single DSCAB with stacks, and to not hinder AGV supplying the production cells, an AGV is placed behind the MBCs. The last MBC in a row is replaced with an MBC slave so the AGV can dock at the back and pick up stacks here. Since the DSCAB only has a docking module in the front, a SPEL is needed at the back of the single DSCAB in order for the AGV to (un)dock from this side. This method of (un)loading the DSCAB from the back means cell demand can be mathed. However, this means the MBCs should be placed next to the DSCAB and not in the wasted space area. This results in identical optimal solutions for BOM and floorspace optimization strategies. Due to this back loading, extra walkways are needed to reach this area, which are included in the total system footprint as well. In figure 7.1 the entire production environment is depicted for this experimental setup and the



(a) Environment

(b) System footprint

Figure 7.1: Generated environment design - experimental setup 6

When further looking in to the AGVs in this experimental setup there are other problems. One DSCAB - Production cell job is defined as: approaching DSCAB, loading, departing, traveling, approaching cell, unloading, departing and traveling back to DSCAB. This amount to a 100 second job for each full, and empty bin. As stated above, the restocking has to be timed just after the bin is emptied to not slow down production. To restock a cell, two AGVs are needed. Since the takt time is 60 seconds, this means only one cell can be restocked every 120 second by one pair of AGVs. Besides the Restocking AGVs, there are two more AGVs needed: at the back of the MBCs and for the finished products.

Since the time of one shift and the bin demand per shift is known, the time between bins per cell can be calculated. This can be seen in table 7.7. To better understand what these times mean, table 7.8 is used. Here intervals are given for each cell, up to 120 minutes. From this table it can be seen that early in the shift there are no problems with resupplying, but slowly more and more cells will need new products. At 120 minutes in to the shift there are 9 different cells requesting a new bin. This would mean there are 18 AGV needed, for just the resupply of the cells.

Product ID	Bins per shift	Time between bins (min)
1	5	96
2	11	44
3	41	12
4	13	37
5	13	37
6	82	6
7	82	6
8	20	24
9	31	16
10	25	20
11	49	10
12	4	120
13	4	120
14	6	80
15	16	30

Table 7.7: Time it takes to empty one bin per cell

Production	Time (min)											
cell ID	6	10	12	16	20	24	30	37	44	80	96	120
1											x	
2									x			
3			x			x		x-			x	x
4								x				
5								x			x	x
6	x		x			x	x	x-			x	x
7	x		x			x	x	x-				
8						x					x	x
9				x						x	x	
10					x					x		x
11		x			x		x			x		x
12												x
13												x
14										x		
15							x					x

Table 7.8: Bin requirement intervals

To find if this experimental setup had any validity, it has to be checked if there is any method of planning the resupply in such a way there are less AGVs needed. This is not the case if each cells is first resupplied when the bin is empty. When allowing for the first bin to be half full on pickup, the entire interval cycle can be shifted. This is essentially an optimization of table 7.8. This optimization can be done through Gurobi as well. Information on time between bins is gathered in the parameter $TimeBbins[i]$. With set i for cell ID ∈ I with length(D) and set j for shift minutes ∈ J with length 480 (one shift) decision variable can be constructed. $Bin_{interval}[i,j]$ tracks each instance of resupply per cell i for minute j. $Interval_{count}[j]$ tracks the cells that are resupplied simultaneously. The following constraints are formulated.

$$\sum_{j=1}^{TimeBbins[i]} Bin_{interval}[i,j] = 1 \qquad \forall i \in I \tag{7.1}$$

The range of sum $\sum_{j=1}^{TimeBbins[i]}$ represents the first bin in the cell. In this time, the model can decide to resupply whenever is optimal, but resupply must have happened before the bins is empty.

$$Bin_{interval}[i,j] = Bin_{interval}[i,j + TimeBbins[i]] \qquad \forall i \in I \qquad \forall j \in J \tag{7.2}$$

Only the first bin in the shift can be switched early. After this first bin, resupplying happens only when the bin is empty. This constraints fills in the rest of the matrix according to the $TimeBbins[i]$.

$$Interval_{count}[j] = \sum_{i=1}^{I} Bin_{interval}[i,j] forjinJ \tag{7.3}$$

Here all resupply occurrences are added together for each minute in the shift. With this last constraint, the objective function can be formulated.

$$MINIMIZE \qquad \max(Interval_{count}[j]) \tag{7.4}$$

Since $Interval_{count}$ indicates the number of resupplies in each minute, the goal is to have no an equal distribution of this function. Even one outlier of 5 resupplies, with the rest being 2 will result in the need of 10 AGVs. In the objective function the maximum value of the $Interval_{count}[j]$ function will be reduced over the entirety of the set J. When plotting function $Interval_{count}[j]$ the resupply instances can be seen. This is done in figure 7.2 for the first two hours of the shift to better see what is happening. It can be seen the maximum number of simultaneous resupplied cells is only two. This brings the need for AGVs for resupply to a total of four. Including the AGVs needed for other activities there are only 6 AGVs needed for this experimental setup, which is almost 25% of the analytically determined amount in table 7.8.

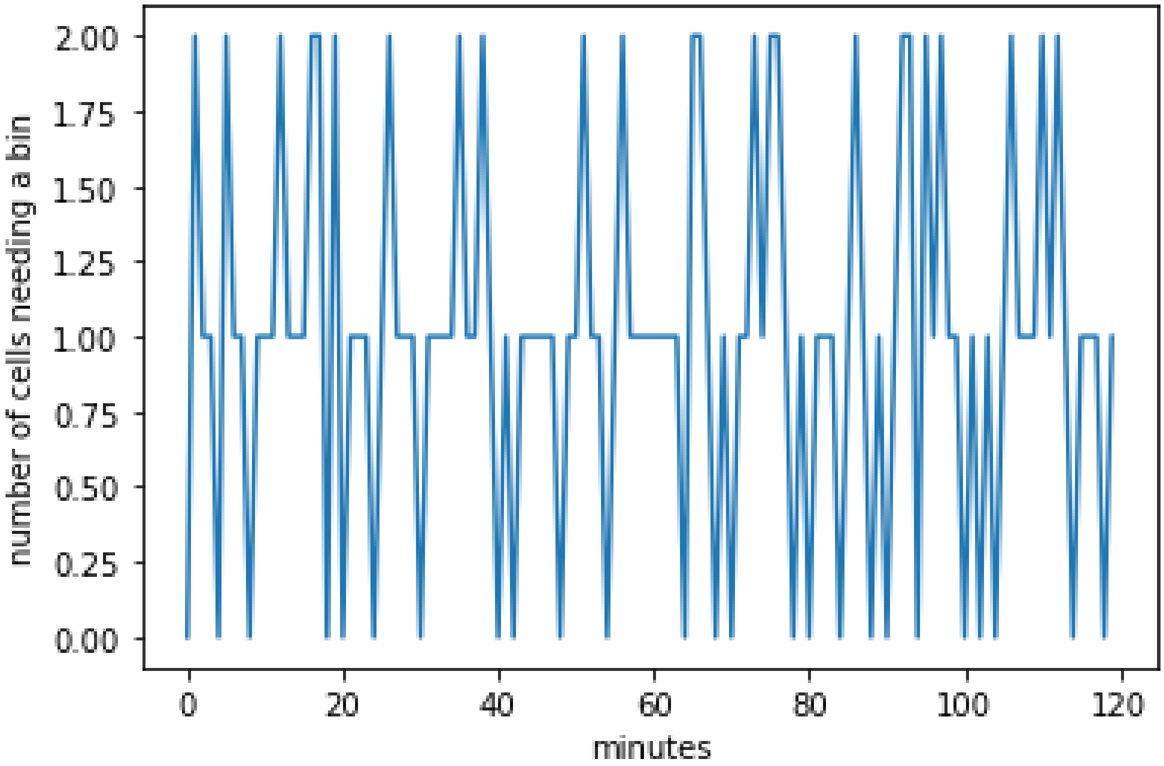


Figure 7.2: Function $Interval_{count}[j]$ for the first two hours of the shift

Now the resupply optimization model is set up, the AGV requirements for different demands can be checked. In table 7.9 the required number of AGVs for different demands are depicted.

Demand modifier	100%	150%	200%	300%	400%	500%	800%	1000%
Max simultaneous resupply	2	3	4	5	5	5	7	9
AGVs needed	6	8	10	12	12	12	16	20

Table 7.9: AGV requirements through resupply timing optimization

7.4. Mobile DSCABs

When adding one directional mobility for the DSCABs, one DSCAB can serve multiple production cells. In figure 7.4 the resulting production environments are illustrated. The alternative positions of the mobile DSCABs are indicated in a lighter shade of orange. The limiting factor for how many cells can be served by one DSCAB is the obstacles in its path. These obstacles are the reservoirs for the liquid used in the product sealing steps of the production and the SPEL at cell C1. Also notable is the two DSCABs at the end of the line. It appears they have no obstacles between them so they can be replaced by one mobile DSCAB. This is not true since the last cell outputs bins of finished products. The DSCAB at cell F feeds empty bins with the correct mold and receives and stacks the full bins. If this DSCAB also has to serve cell 15, which outputs empty bins, there would be mixed stacks of empty bins and finished product bins. In table 7.10 the results of this experimental setup are depicted. Here it can be seen that however the number of units required is drastically lower, the BOM costs are still at 83 %. This can be explained by the high costs of the mobile DSCABs. Since there are obstacles which the DSCABs cannot pass, there is room for MBCs in the wasted space area.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	87%	83%	79%	78%	98%	108,51
Floorspace	87%	82%	79%	78%	101%	100,11

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**	**Mobile DSCAB**
BOM	4	7	14	0	4	1	4
Floorspace	4	10	11	0	4	1	4

Table 7.10: Results of the mobile DSCABs experimental setup w.r.t. the case study

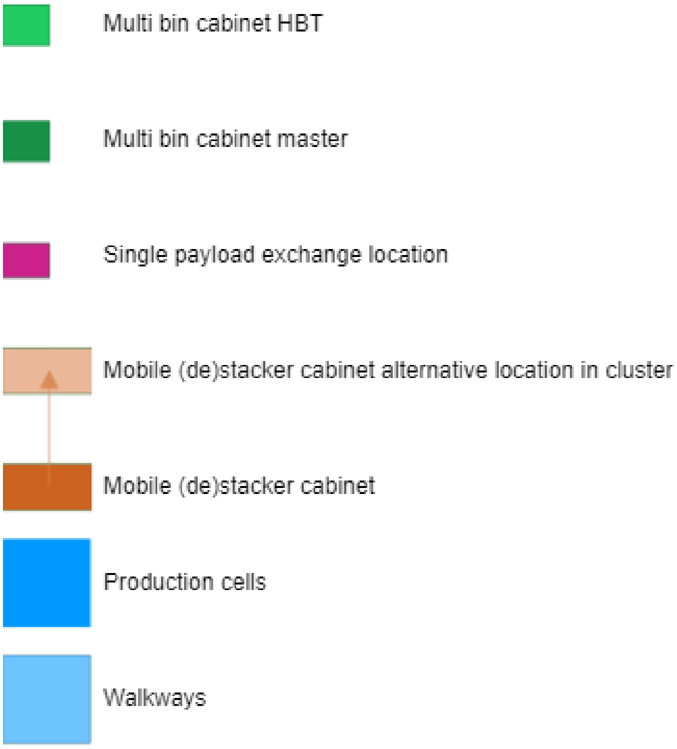
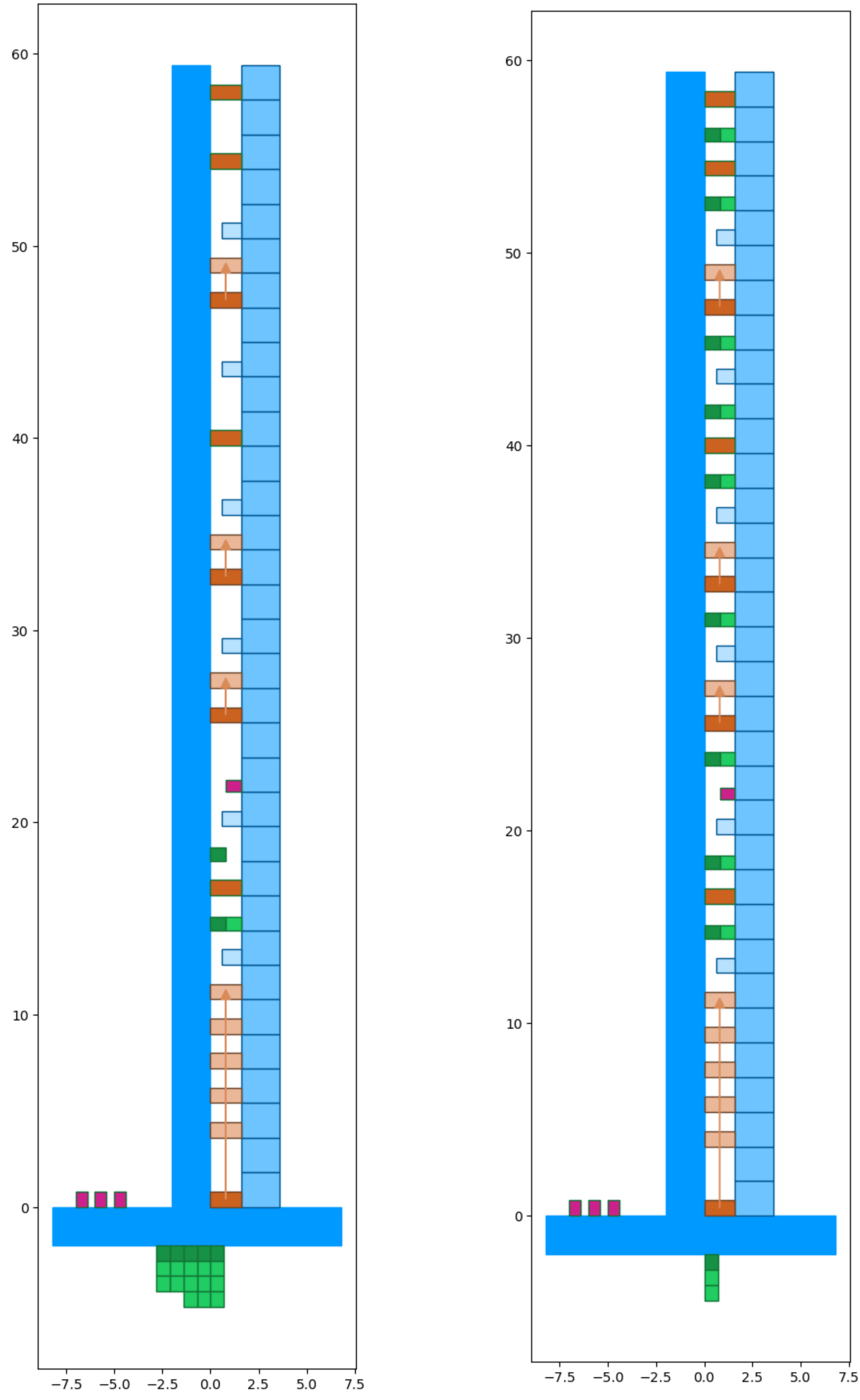

Multi bin cabinet HBT
Multi bin cabinet master
Single payload exchange location
Mobile (de)stacker cabinet alternative location in cluster
Mobile (de)stacker cabinet
Production cells
Walkways

Figure 7.3: Legend for figure 7.4



(a) BOM optimal solution

(b) Floorspace optimal solution

Figure 7.4: Generated environment designs - experimental setup 7

7.4.1. No obstacles

When not encountering obstacles, the DSCAB could theoretically serve every cell. In practice, this would mean it has to be very very fast, both in movement speed and other actions. To determine the maximum amount of cells that can be served by one mobile DSCAB the new limiting factor must be identified. This is thought to be time. The bin demand of the cell with the highest demand dictates the time the DSCAB has to serve other cells. In order to determine the time a DSCAB needs to serve all cells, table 7.11 is needed. Here the time time between bins and distance between following cells is depicted.

Cell ID	Bins per shift	Time between bins	Distance between cells
1	82	6	
			4
2	41	12	
			2
3	31	16	
			2
4	13	37	
			2
5	5	96	
			2
6	13	37	
			6
7	11	44	
			10
8	4	120	
			2
9	4	120	
			6
10	6	80	
			2
11	25	20	
			6
12	20	24	
			8
13	82	6	
			2
14	16	30	
			6
15	49	10	

Table 7.11: Bin demand per cell in order

The assumption is made that there is the possibility that all cells in a cluster (cluster of all cells served by one DSCAB) have to be served in one cycle. Cycle time is dictated by the cell with the highest demand. When selecting cells 1 to 5 as a cluster, cell 1 would be the this cell. This cell we will call P_c for *production cell critical* and is seen as the "base" for the DSCAB. The total distance the DSCAB may have to cover in a cycle is twice the distance from the beginning to the end of the cluster. This distance we will denote as $\sum_{n=1}^{N-1} D_{nn+1}$. Here n is the the number of cell in the cluster. The time is takes the DSCAB to carry out all actions also has to be determined. One action is summarized in variable $Y' = [approachtime, (un)loadtime, departtime]$. v denotes the speed of the DSCAB and is estimated to be 1 m/s. Once the time of one action is known, the total time of a cycle can be determined by multiplying by the number of cells in the cluster. This results in the equation 7.5.

$$Y_t = 2 * \sum_{n=1}^{N-1} D_{nn+1} * \frac{1}{v} + n * \sum Y' \tag{7.5}$$

Here Y_t represents the total action time of a cycle. This total action time should not exceed the cycle time. If this happens, the DSCAB can not be back in time to start the next cycle. Restocking by an AGV and driving back to base should also be considered. Driving back to the P_c cell is assumed to always be equal to the maximum posible travel time $\frac{D_{max}}{v}$. Restocking is dictated by the AGV since the DSCAB need to be static for this period. Restocking and unloading in to a cell can not happen at the same time. From section 4.1 the restocking time can be determined. Since one restocking action can restock one single stack with a maximum of 7 bins, exceeding more than 7 cell in a cluster requires an extra restocking action. One restocking action is summarized in Y_r'. The time is takes to restock can therefore be formulated as stated in equation 7.6.

$$T_{restock} = \mathit{roundup}(n/7) * Y_r' \tag{7.6}$$

$$Y_t \leq S_c - T_{restock} - \frac{D_{max}}{v} \tag{7.7}$$

Equation 7.7 shows the constraint which must be full filled when assigning clusters in order for a cluster to be viable. Here S_c represents the time between bins as demanded by the P_c cell. Using the equations, clusters of cells can be made. This is shown in table 7.12. For these calculation Y' is estimated at $[5, 15, 5]$

Cluster 1	From	1	ID
	to	8	ID
	Pc	1	ID
	Sc	6	min
	Dmax	28	m
	Davg	3,5	m
	#	8	quantity
	Trestock	0,83	min
	Y’	25	sec
	Yt	4,3	min
	Viable	YES	

Cluster 2	From	9	ID
	to	13	ID
	Pc	15	ID
	Sc	6	min
	Dmax	30	m
	Davg	4,3	m
	#	7	quantity
	Trestock	0	sec
	Y’	25	sec
	Yt	3,9	min
	Viable	YES	

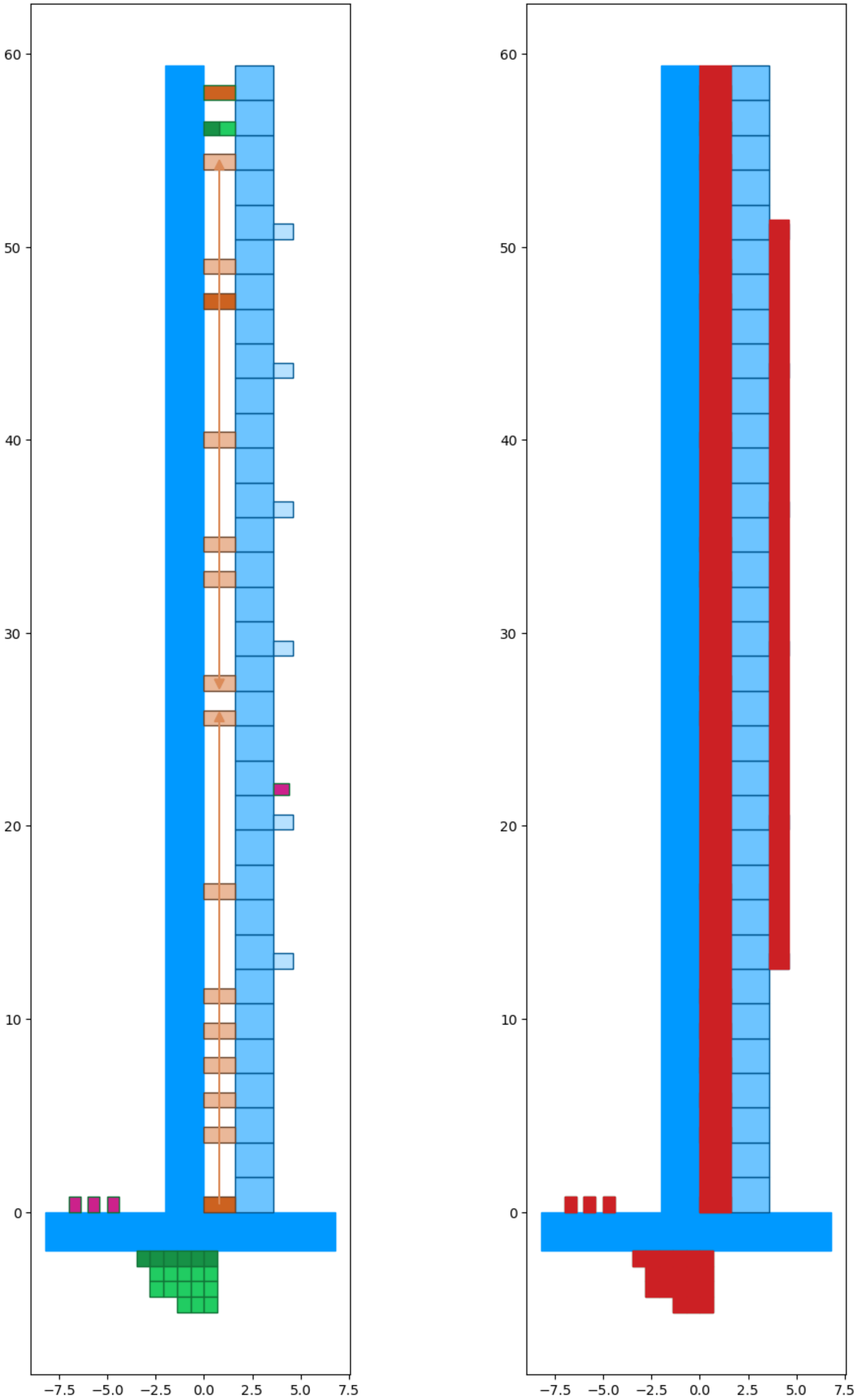
Table 7.12: Cluster selection according to equations 7.5 and 7.7

When implementing these clusters in our model, the results in table 7.13 are obtained. Since there is no space for MBCs in the wasted space area, the optimal solutions for both BOM and floorspace are identical. The resulting environment is depicted in figure 7.5. Here it can be seen how large the clusters can be. Since the obstacles need to be moved to the other side of the production line, the system footprint increases quite heavily.

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	76%	52%	66%	65%	135%	149,07
Floorspace	76%	51%	65%	64%	151%	149,07

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**	**Mobile DSCAB**
BOM	4	7	14	0	1	1	2
Floorspace	4	7	14	0	1	1	2

Table 7.13: Results of the mobile DSCABs without obstacles experimental setup w.r.t. the case study



(a) Environment

(b) System footprint

Figure 7.5: Generated environment design - experimental setup 8

7.5. AGV with (de)stacker function

When outfitting the AGVs with a stacker - destacker module, the DSCABs in front of the production cells become obsolete. Instead, a SPEL is needed since the cells themselves do not have the docking capabilities required for connecting with an AGV. To determine the number of AGVs needed, first the limiting factor has to be determined. Supplying all the cell in time is this limiting factor since only one bin can be present in each cell. This problem is similar to the problem discussed in section 7.4.1. Here clustering is also key to finding the required number of AGVs.

One cycle is now defined as traveling from MBC to the cluster and back, and traveling from cell to cell. Important is to note a cycle has a maximum of 7 cell since the AGV can only carry one stack and one stack is at most 7 bins high. Traveling to and from the cluster is assumed to have an average distance of $(\sum_{i=1}^{I} D_{mc,i})/I$ where I is the number of cells in the cluster. Also, approach, (un)load and depart time should be considered. These are included in variable Y'_{mc} as $[approachtime, (un)loadtime, departtime]$. These actions only occur once per cycle in the MBC to cell part of the cycle. The approaching, loading and departure times of each cell are included in the cell-to-cell part of the cycle. Now an expression for the variable Y_{mc} can be formulated.

$$Y_{mc} = \sum Y'_{mc} + 2 * \sum_{i=1}^{I} D_{mc,i} * \frac{1}{I * v} \tag{7.8}$$

Next an expression for the cell to cell actions, Y_{cc} is required. The total distance covered in a cluster is easily calculated by $\sum_{i=1}^{I} D_{cc,ii+1}$. Here approach, (un)load, and depart time are also included in variable Y'_{cc}. Now an expression can be found for Y_{cc} and eventually for total cycle time Yt. This total cycle time can not exceed the time between bin demand of the most demanding cell in the cluster. This time is denoted by S_c standing for critical requested supply.

$$Y_{cc} = I * \sum Y'_{cc} + \sum_{i=1}^{I} D_{cc,ii+1} * \frac{1}{v} \tag{7.9}$$

$$Y_t = Y_{mc} + Y_{cc} \leq S_c \tag{7.10}$$

Cluster 1	From	1	ID
	to	7	ID
	n	7	quantity
	Dmc, avg	10	m
	Dcc, tot	18	m
	Sx	6	min
	Ymc	50	sec
	Ycc	236	sec
	Yt	4,8	min
	Viable	YES	

Cluster 2	From	8	ID
	to	12	ID
	n	5	quantity
	Dmc, avg	37	m
	Dcc, tot	16	m
	Sx	20	min
	Ymc	88	sec
	Ycc	173	sec
	Yt	4,4	min
	Viable	YES	

Cluster 3	From	13	ID
	to	15	ID
	n	3	quantity
	Dmc, avg	57	m
	Dcc, tot	8	m
	Sx	6	min
	Ymc	117	sec
	Ycc	101	sec
	Yt	3,6	min
	Viable	YES	

Optimization	**Total**	**BOM**	**Assembly**	**Installation**	**Floorspace**	**Floorspace (m2)**
BOM	74%	63%	94%	85%	100%	110,73
Floorspace	74%	64%	94%	86%	101%	99,53

Optimization	**SPEL**	**MBC master**	**MBC HBT**	**MBC slave**	**DSCAB**	**AGVs**	**AGV (de)stackers**
BOM	19	7	14	0	1	0	3
Floorspace	19	11	10	0	1	0	3

7.6. Takt time modification

One of the KPIs this thesis test is the ability of a bin handling system to cope with demand growth. Since the demand is defined by the production line, it is directly linked to the takt time of the products. A shorter takt time means more products will be produced per hour, and more semi products are needed by the individual production cells. This increase in demand will require more storage space and handling speed from the bin handling system. Three main factors are identified which are likely to drive up the costs when increasing demand: storage space, AGV requirement and DSCAB supply rate. In this section all experimental setups are testes for different product takt times in order to how the system costs and footprint will change. Extended results are depicted in appendix C.

One important thing to note is experimental setup 6 - the central DSCAB. As stated in section 7.3, this experimental setup is bound to a maximum takt time of 40 seconds due to the operation speed of the AGV approach, depart and (un)load time. To be able to compare results, this restriction is ignored and the assumption is made the AGV action time can cope with the takt time.

7.6.1. Footprint optimal solutions

In table 7.14 the changes in system footprint for each experimental setup are depicted for the floorspace optimal solutions. Here it can be seen most experimental setups increase in a similar way, except experimental setup 3. The system footprint of this experimental setup only increases once the takt time is 35 or lower. This means experimental setup 3 is most fitted for an environment with a growing demand, when prioritizing floorspace.

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-0,6	0,0	0,0	-0,6	-0,6	-1,1	-0,6	-0,6	-0,6
60	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
50	2,8	1,1	0,0	2,8	2,8	4,2	3,9	2,8	2,8
43	4,5	2,8	0,0	4,5	5,0	5,9	4,5	4,5	4,5
35	7,8	6,7	3,4	7,8	8,4	9,3	7,8	7,8	7,8
33	10,5	10,0	5,5	10,5	11,1	10,9	10,5	10,5	9,5
30	11,6	10,5	8,3	12,2	12,8	12,1	11,6	11,6	10,6

Table 7.14: System footprint increase in m^2 w.r.t. to takt time of 60 seconds

The total costs for the different takt times are also determined. Experimental setups might have a small increase in footprint, but a large increase in total costs. In table 7.15 these results are depicted as a percentage of the total system costs at a takt time of 60 seconds. The largest increases are in experimental setups 7 and 8. Not coincidentally these experimental setups are variant of each other. The jumps in total costs between a takt time of 35, 33 and 30 seconds can be tracked to the need for an extra mobile DSCAB. The clusters as described in section 7.4.1 can not cope with this demand and extra DSCABs are required. This is an example of a failing DSCAB supply rate. When tracking results, it is concluded experimental setups one though 5 mainly cope with an increase in storage capacity, which in turn increase the total costs. At one point, at a takt time of 33 seconds, one AGV cannot handle the demand on its own and an additional AGV is required.

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	0%	-1%	-5%	-2%	-1%	-1%	-2%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	5%	4%	5%	5%	12%	9%	14%	7%
43	8%	9%	6%	8%	9%	17%	10%	19%	11%
35	14%	16%	12%	14%	15%	24%	24%	26%	19%
33	20%	23%	17%	20%	21%	28%	31%	34%	23%
30	22%	24%	22%	23%	25%	32%	33%	37%	25%

Table 7.15: Total costs increase in w.r.t. to takt time of 60 seconds

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-2%	-2%	0%	-2%	-7%	-3%	-1%	-2%	-3%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	6%	5%	5%	5%	18%	10%	27%	8%
43	9%	10%	8%	9%	10%	25%	11%	33%	14%
35	15%	17%	14%	15%	16%	35%	29%	45%	23%
33	22%	26%	19%	22%	23%	41%	38%	59%	28%
30	24%	27%	24%	25%	27%	46%	40%	64%	31%

Table 7.16: BOM costs increase in w.r.t. to takt time of 60 seconds

7.6.2. BOM optimal solutions

As with the footprint optimal solutions, the increase in system footprint can be checked for the BOM optimal solutions. These results are depicted in table 7.17. Since the BOM optimal solution prefers to place storage units in rows as long as possible, most storage units will be placed outside of the wasted space area. The ability to place units inside of the wasted space area therefore does not play a big role anymore. It can be seen how the increase in floorspace is comparable in all experimental setups.

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-0,6	-0,6	-0,6	-0,6	-0,6	-1,1	0,0	-0,6	-0,6
60	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
50	2,8	2,8	1,7	2,8	2,2	4,2	2,2	2,8	2,8
43	4,5	4,5	3,4	4,5	3,9	5,9	4,5	4,5	4,5
35	7,8	8,4	6,7	7,8	6,2	9,3	7,3	7,8	7,8
33	10,5	11,6	8,8	10,5	8,8	10,9	10,5	10,5	9,5
30	11,6	12,2	11,6	12,2	10,0	12,1	11,1	11,6	10,6

Table 7.17: System footprint increase in m^2 w.r.t. to takt time of 60 seconds

In tables 7.18 and 7.19 the increase in total costs and BOM costs are depicted for the BOM optimal solutions of different takt times.

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	-1%	-1%	-1%	-2%	-1%	-1%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	5%	3%	5%	4%	12%	5%	14%	7%
43	8%	8%	6%	8%	6%	17%	9%	19%	11%
35	14%	15%	12%	14%	10%	24%	23%	26%	19%
33	20%	23%	16%	20%	16%	28%	30%	34%	23%
30	22%	24%	22%	23%	18%	32%	32%	37%	26%

Table 7.18: Total costs increase in w.r.t. to takt time of 60 seconds

	Experimental setup								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	-2%	-2%	-1%	-3%	-1%	-2%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	6%	3%	5%	4%	18%	6%	27%	9%
43	9%	9%	6%	9%	7%	25%	11%	33%	14%
35	15%	17%	13%	15%	11%	35%	29%	45%	24%
33	23%	26%	18%	23%	18%	41%	38%	59%	30%
30	25%	27%	23%	26%	20%	46%	41%	64%	34%

Table 7.19: BOM costs increase in w.r.t. to takt time of 60 seconds

7.7. Chapter summary

In this chapter the results of each individual experimental setup are discussed, as well as how they react to different takt times. The goal of this chapter is to find an answer to the subquestion: **'What is the impact of different design strategies?'.** To recap, the results are gathered once again in tables 7.20 and 7.21.

In experimental setups 2 to 5 only small variations are made on the already existing system. In experimental setup 2 the existing storage locations are used in order to reduce the required specialised storage units, the MBCs. Through this relatively simple solution of coordinating and implementing more thoughtfully 4% can be saved on the total system costs and 2 % on system footprint. Taking this experimental setup further in experimental setup 3, the benefits of requiring less MBCs do not outweigh the extra costs of the DSCABs anymore. Here we see an increase in costs of 6%. The necessity of having storage space for 3 stacks in each DSCAB can be argued. From table C.2 in appendix C it can be concluded some DSCABs only use one or two stacks in an entire shift. This results in empty but costly storage locations in the system. Since the DSCABs are required for (de)stacking, they are already there. Therefore it still is a viable method of reducing system footprint.

One seemingly effective method of reducing the number of MBCs required by the system was the shared storage strategy from experimental setup 4. When reasoning on this strategy one can get the impression storing multiple products next to each other reduces the number of empty storage locations. In reality the number of empty storage locations are minimal to begin with. For the specific demand, both quantity and variety, the shared storage strategy is almost equivalent to the case study optimal solution. Experimental setup 4 is bounded by the geometry of the production environment. In experimental setup 5 this restriction is lifted. When allowing for placement anywhere, one long row of MBCs is formed. This is due to the MBC HBTs being cheaper. This is only the case when optimizing for BOM since the footprint optimal solutions opts for placement in the wasted space area.

More divergent experimental setups are experimental setups 6 to 9. Here units are redesigned. In experimental setup 6 the idea of having just one DSCAB is explored. Since DSCABs are the most expensive and abundant units in the system, saving costs here could potentially have a great impact. Transporting bins instead of stack increases the AGV requirements, but drastically decreases the BOM of the system. Through planning optimization the minimum number of AGVs required can be reduced but then another factor is introduced: transporting half-full bins. While this is not a problem on its own, it should be noted since it will require additional actions to sort out these bins. Once takt time decreases, this experimental setup becomes notably less efficient. Due to more AGVs the BOM costs increase next to the increase in BOM due to the growth in MBC units. Restocking cells now requires two AGVs. This restocking takes 40 seconds, which puts a hard limit on the possible takt time of the production line. The takt time of 30 seconds therefore is purely theoretical. This experimental setup also required a notable larger footprint.

Takt time - 60s	Experimental setup								
	1	2	3	4	5	6	7	8	9
Total costs	100%	96%	106%	100%	100%	85%	88%	76%	74%
BOM costs	100%	95%	110%	100%	100%	67%	82%	51%	64%
Floorspace	100%	99%	99%	100%	100%	136%	106%	151%	101%
Floorspace m2	99,0	98,4	98,4	99,0	99,0	134,8	104,6	149,1	99,5

Takt time - 30s	Experimental setup								
	1	2	3	4	5	6	7	8	9
Total costs	100%	98%	106%	101%	102%	92%	96%	86%	76%
BOM costs	100%	98%	111%	101%	103%	79%	93%	68%	68%
Floorspace	100%	98%	96%	101%	101%	133%	106%	145%	100%
Floorspace m2	110,6	108,9	106,7	111,2	111,7	146,9	117,3	160,7	110,2

Table 7.20: Summary of results w.r.t case study - floorspace optimal solutions

Takt time - 60s	Experimental setup								
	1	2	3	4	5	6	7	8	9
Total costs	100%	96%	105%	99%	94%	85%	88%	76%	74%
BOM costs	100%	96%	111%	99%	92%	69%	83%	52%	63%
Floorspace	100%	98%	96%	100%	98%	122%	103%	135%	100%
Floorspace m2	110,2	108,0	105,7	110,2	108,0	134,8	113,0	149,1	110,7

Takt time - 30s	Experimental setup								
	1	2	3	4	5	6	7	8	9
Total costs	100%	97%	105%	100%	91%	91%	95%	85%	77%
BOM costs	100%	97%	109%	100%	89%	80%	93%	69%	67%
Floorspace	100%	99%	96%	100%	97%	121%	103%	132%	100%
Floorspace m2	121,8	120,1	117,3	122,4	117,9	146,9	125,2	160,7	121,4

Table 7.21: Summary of results w.r.t. case study - BOM optimal solutions

In experimental setup 7 another variation of the DSCAB is introduced: the mobile DSCAB. By making this unit mobile in one direction, the DSCABs which are idle for most of the shift can be eliminated. When looking at the generated environment design for this experimental setup it is clear there are obstacles in its path. While saving on costs, the amount of savings rely heavily on the obstacles around the production line. To test the influence of the obstacles, experimental setup 8 is introduced. Here all the obstacles are moved to the other side of the production line. This drastically increases the floorspace required by the system. Also, since no obstacles next to the line are allowed, no MBCs can be placed in the wasted space area, increasing the system footprint even further. This comes with the savings of almost half of the BOM costs of the system. When increasing the demand, the BOM savings reduce but are still notable.

Lastly, in experimental setup 9 the DSCAB capabilities are taken even one step further. Instead of unidirectional movement, the DSCABs now have the movement of an AGV. Or the other way around, the AGVs are outfitted with the (de)stacker modules. This experimental setup cuts down BOM costs by being able to replace all DSCABs with SPELs. At the same time, only three AGVs are required even for a takt time of 30 seconds.

In table 7.22 The impact of all experimental setups is summarized. For visualization purposes, the following classifications are used:

Minimal	0 - 2 %	Moderate	8 - 15 %	Large	26 - 35 %
Small	3 - 7 %	Substantial	16 - 25 %	Massive	36 - 50 %

Experimental setup	Direct impact	Indirect impact
2	Small total costs savings	Minor planning changes
	Small BOM savings	
	Minimal floorspace savings	
	Results steady through demand increase	
3	Small total costs Increase	Minor planning changes
	Small BOM increase	Moderate unit modification
	Small floorspace savings	
	Results steady through demand increase	
4	Minimal total costs savings	Moderate planning changes
	Minimal BOM savings	Moderate unit modification (sensor placement)
	No floorspace savings	
	Results steady through demand increase	
5	Small total costs savings	Environment requirements
	Small BOM savings	Moderate planning changes
	Minimal floorspace savings	Moderate unit modification (sensor placement)
	Results steady through demand increase	
6	Moderate total costs savings	Large unit modification
	Large BOM savings	Large planning changes
	Large floorspace increase	Bin sorting post-production
	Less effective through demand increase	Bin supply order required
		Walkway enlargement
7	Moderate total costs savings	Large unit modification
	Substantial BOM savings	Large planning changes
	Small floorspace increase	Bin supply order required
	Less effective through demand increase	
8	Substantial total costs savings	Large unit modification
	Massive BOM savings	Large planning changes
	Masive floorspace increase	Bin supply order required
	Less effective through demand increase	
9	Large total costs savings	Large unit modification
	Large BOM savings	Large planning changes
	Minimal floorspace increase	Bin supply order required
	Results steady through demand increase	

Table 7.22: Overview of impact of each experimental setup

		Experimental setup								
	Takt time (s)	1	2	3	4	5	6	7	8	9
Total costs	60	100%	96%	106%	100%	100%	85%	88%	76%	74%
	30	100%	98%	106%	101%	102%	92%	96%	86%	76%
BOM costs	60	100%	95%	110%	100%	100%	67%	82%	51%	64%
	30	100%	98%	111%	101%	103%	79%	93%	68%	68%
Floorspace	60	100%	99%	99%	100%	100%	136%	106%	151%	101%
	30	100%	98%	96%	101%	101%	133%	106%	145%	100%
Floorspace m2	60	99,0	98,4	98,4	99,0	99,0	134,8	104,6	149,1	99,5
	30	110,6	108,9	106,7	111,2	111,7	146,9	117,3	160,7	110,2

Table 7.23: Floorspace

		Experimental setup								
	Takt time (s)	1	2	3	4	5	6	7	8	9
Total costs	60	100%	96%	105%	99%	94%	85%	88%	76%	74%
	30	100%	97%	105%	100%	91%	91%	95%	85%	77%
BOM costs	60	100%	96%	111%	99%	92%	69%	83%	52%	63%
	30	100%	97%	109%	100%	89%	80%	93%	69%	67%
Floorspace	60	100%	98%	96%	100%	98%	122%	103%	135%	100%
	30	100%	99%	96%	100%	97%	121%	103%	132%	100%
Floorspace m2	60	110,2	108,0	105,7	110,2	108,0	134,8	113,0	149,1	110,7
	30	121,8	120,1	117,3	122,4	117,9	146,9	125,2	160,7	121,4

Table 7.24: BOM

8
Discussion

This thesis aims to answer the question: **what are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?** This is done by first constructing a method of evaluating the performance of a bin handling system. Through a deterministic linear optimization model, the optimal configuration of a bin handling system is determined. Now a method of generating optimal solutions is constructed, the performance of different bin handling systems can be compared.

To do this, different experimental setups are generated. In the first few experimental setups there is a focus on the implementation of the system and small changes to existing strategies. Here it is found that small storage capacities at each cell, instead of more general storage capacity, has a positive effect on the total BOM. Due to this potential, the storage capacities at each cell have been expanded. Such a strategy turned out to have a negative impact on the BOM, while having a small positive impact on the floorspace. Having this expanded storage space at each cell, some storage location will remain empty due to the low bin requirement of the cell.

In order to maximize the use of the unique storage locations, a shared storage strategy is introduced. Shared storage strategies are common in warehouses due to ability of always being able to fill an empty spot. In a bin handling system such as evaluated in this thesis, the storage stage does not have many empty slots to begin with. The demand of a production line is known far in advance and does seldom fluctuate. A shared storage strategy turned out not to be effective in both reducing footprint and BOM. At lower demands, shared storage strategies might have more of an effect since then storage units are configured differently and shared storage enables more configurations. When designing the production environment around a shared strategy, i.e. if the designer has all freedom, the BOM can be notably reduced.

Transporting bins, instead of stacks of bins which need to be destacked at each cell, introduces new obstacles. It is has been proven the number AGVs needed in such an experimental setup counter intuitively does not increase as much as would have thought. By optimizing resupply strategies, it is possible to maintain a reasonable number of AGVs required for efficient operations. The number of (de)stackers reduces drastically, and so does the BOM. This happens at a trade-off in floorspace, which increases. However, the effectiveness of this experimental setup drops when demand increases. Transporting bins has one big drawback: the action time of the AGVs. The product takt time is limited by the AGVs capabilities to extract and insert bins into the cell fast enough.

The possibility of unidirectional, mobile (de)stackers is also explored. While having a comparable system footprint as the case study configuration, BOM costs go down. This happens even though the new (de)stackers are more expensive and there are obstacles in their paths. By removing the obstacles, the total BOM costs drop even further. Since obstacles are moved, the total system footprint increases drastically. When demand increases, the effectiveness of such a strategy drops since more of the extra expensive (de)stackers are needed. From these experimental setups it can be concluded that due to the (de)stackers high BOM, lowering the number needed will have a large impact on the costs. By taking the concept of a mobile (de)stackers one step further, a strategy of outfitting the AGVs with a (de)stacker module is tested. Since they can move around obstacles the total system footprint does not increase. The BOM costs drop drastically since there are no more expensive static (de)stackers. In contrast to the unidirectional (de)stackers, results are now stable when increasing demand.

This chapter also reviews the factors that may potentially impact the findings of the research, through a detailed discussion. The first of which is the configuration of the production environment. A bin handling system is only a small part of the total production environment. Production line configuration is defined as a constant, which means the model constructed in this thesis cannot alter it in order to improve results. This is done in order to be able to compare results of different experimental setups more accurately. It is possible that certain experimental setups or design solutions may exhibit better performance on particular production line configurations, such as a U-shaped line. A U-shaped line is taken as example here, but there are infinitely more production line configurations. Included in the production line configuration is the requirement of liquid reservoirs or obstacles next to the line. As can be seen in experimental setup 7, these play a big role in the effectiveness of some experimental setups.

As previously stated in section 3.7.2, there are several ways to define the system footprint.In this thesis the choice is made to include space which is made unusable by the system in the total system footprint. However, the level to which space is unusable can differ. As mentioned in section 4.2.1, there can be more factors restricting usage of space in and around the system. The number, and level of contribution, of different factors can differ from situation to situation. The depreciation costs also contribute directly to the floorspace costs, and thus to the optimal generated floorspace design. As floorspace gets more expensive, the optimal solution will place more an more units in the wasted space. Therefore these optimal solutions are situational.

In this thesis, the bin handling system was considered to have an infinite and instant supply of new products. While this is often the case in production environments, there are situations where supply is not steady. This should be considered when interpreting the results of this thesis.

9
Conclusions

This thesis aims to inform the reader on the performance of different design choices of autonomous bin handling systems. In section 9.1 the findings of this research are summarized. Lastly, in section 9.2 recommendations on future work are given.

9.1. Conclusions

Autonomous, flexible bin handling systems are state of the art. Designing such systems therefore is often done with few examples or knowledge of performance alteration design strategies. This thesis composed a quantitative optimization model which is used on different design strategies of bin handling systems.This is done in order to find an answer on the main research question: **what are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?** Design strategies researched in this thesis are compared to the case study system. Conclusion are drawn in reference to performance of the optimal solutions for the case study system

Firstly, with the definition of system footprint this thesis proposes, only small reductions of 2 to 4 % in footprint are found. Reductions found are mainly caused by the placement of units in bill of material wise undesirable locations. A second factor found for the reduction of systems footprint is the extended use of local buffers. Local buffers are located in the already existing equipment and thus do not increase system footprint. In contrast to warehousing, shared storage strategies have minimal impact on both costs and floorspace in bin handling systems. Design choices such as mobile (de)stacker equipment and bin transportation instead of stack transportation proved to have a negative impact of floorspace. Mobile (de)stackers require the relocation of obstacles, or other equipment, to reach their full potential. This increases the system footprint drastically by 35 %. The transportation of bin instead of stacks introduced more traffic in the environment. This traffic leads to the need for wider walkways and thus an increase in footprint. This increase can amount to up to 22 %.

Secondly, design strategies are evaluated for their bill of material costs. While extended local buffers have a positive influence on floorspace, the system BOM increases by 10 %. The individual BOM of existing units increases due to the requirement a larger internal storage location, which ultimately increases the BOM. Shared storage strategies, which are common in warehouse optimizations, are also tested. Due to limitations on available space in the production environment there is only a 1% improvement in BOM. When removing space limitations, this improvement grows up to 8 %. While increasing system footprint, bin transportation instead of stack transportation decreases the BOM by 30 %. This seems counterintuitive since transporting bin would seem to lead to the need for an excessive amount of transporting units. This reduction is due to the elimination of most (de)stacker units, and a quantitative optimal solution for resupply planning. The largest savings in BOM are found when adding mobility functions to (de)stacker equipment. Here a decrease of 48% can reached. When fully integrating the transportation and (de)stacking function in one unit, a positive result for the system BOM is found. This strategy does not seem to have a negative result for the system footprint. A reduction of 36 % is obtained when combining these function in one unit, while keeping system footprint constant with the case study.

Lastly, the throughput capabilities of the design strategies are evaluated. Product takt time is decreased and the changes in results are compared. Noteworthy is the the improved result for shared storage without placement limitations. Here the total system costs reduced further from 6% at a takt time of 60 seconds to 9% at a takt time of 30 seconds. Effectiveness of solutions as bin transport instead of stack transport and mobile (de)stackers reduces as takt times get lower. Bin transport strategies see the total cost reduction of 15 % at a takt time of 60 seconds go to 9% reduction at a takt time of 30 seconds. Mobile (de)stackers have their total cost reduction reduce to 15 % at a takt time of 30 seconds. For this strategy, total cost reduction was 25 % at a takt time 60 seconds. Full integration of the transportation and (de)stacking function shows only a 3% less effective total cost at a 30 seconds takt time. Floorspace still does not increase in this strategy.

9.2. Future research

This thesis only touches the hardware design part of bin handling systems. Bin handling systems are part of a larger system, the production environment. In order to keep results reliable, other factors are kept constant since their influence could not reliably be estimated and quantified. For future research promising experimental setups from this thesis might be used and other factors can be altered such as:

Production line layout. Placement of bin handling units is heavily restricted in this thesis due to the predefined production line placement. The production line is defined as a line. However, production lines come in all kinds of shapes and configurations. Testing the validity of experimental setups given in this thesis for different production line configurations will broaden the understanding of certain design choices.

Not only the predefined production line layout but also the production environment layout limits the placement of bin handling units. Future work might consider total freedom in unit placement by having no restrictions on production environments.

This study is based on a case study. This case study has such a throughput that the employed AGVs are able to cope with demand easily. Therefore unit placement optimization for AGV performance is left outside of the scope. For future work, when assessing higher throughput systems this should be taken into account in order to find the absolute minimal number of AGVs required.

Lastly, in this thesis bin type, shape and size are based on the bins already existing in the case study. Bins play a big role in the performance of bin handling systems. Further research could look into the alteration of the bins in order to increase performance. Since bins are supposed to be universal, meaning all products should fit in them and all units should be able to handle them, all factors surrounding the bins should be considered. This thesis considers two types of bins. Future works could also consider more, or less types.

91

A

Research Paper

The effects of design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?

Tim Maaskant Ir. L. de Wit Ir. K. van den Elsen
Dr. W.W.A. Beelaerts van Blokland F. Schulte

April 21, 2023

Abstract

Fully autonomous bin handling systems are relatively new and examples of existing systems are hard to find. This makes designing such a system more difficult. The goal of this research is to provide insight in the effects of different design strategies of autonomous bin handling systems. This is done through a quantitative optimization model of an existing autonomous bin handling system. With this model, modifications to the system are made and evaluated. Optimal solutions for bill of material and floorspace, for different demand quantities are generated and compared.

Keywords: Bin handling system, AGV, Production environment, Generative design, Quantitative modeling, Gurobi Optimizer, System design, Manufacturing support systems

1 Introduction

Throughout history, people have striven for efficiency. This is especially true for manufacturing processes. With inventions such as the steam engine and the conveyor belt production could be increased drastically. Nowadays most manufacturing processes are highly automated and optimized, Jayal et al., 2010. The remaining part in the automation process is the logistics, more specifically the intralogistics, Wang et al., 2009. The implementation of autonomous transport for goods in a factory presents both mechanical and logistical challenges. Since goods can greatly vary in size and shape, equipment used to handle them must possess a wide range of mechanical properties, Kroemer, 2017. By introducing a standardized transport bin, equipment can be simplified. A system designed to handle standardized bins is referred to as a bin handling system.

Autonomous bin handling systems are relatively new, and there are few existing examples of such systems. Designing such a system is therefore difficult. The effects and impacts of certain design choices may only become evident later in the development process. According to Baral, 2021, up to 50% of the costs associated with a product are incurred during the design conceptualization phase of the development process. Therefore, cost reduction becomes increasingly difficult in later stages of the process. Being able to make informed, well grounded decisions during this phase is therefore crucial. This paper aims to provide insight in the impact of different design choices, better informed decisions can be made when developing, or altering bin handling systems. The research questions therefore is:

What are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?

This research includes a case study of a real life bin handling system present at Prodrive Technologies, a company specialized in high tech industrial automation solutions. This case study is used as a benchmark to evaluate the performance of other systems.

First, existing storage strategies and bin handling systems are evaluated in section 2. The case study systems is further identified in section 3

Next, a mathematical model of the bin handling system is constructed. This model is then used in a computational solver to find a quantitative optimal solution based on costs or floorspace in section 4. Here the product takt time is varied to mimic an increase in demand and changes in optimal solutions are evaluated. Scenarios of different bin handling systems are constructed and tested in the same method. Lastly, in section 5, conclusions are drawn based on the findings, and recommendations for future work are made.

2 Literature and methodology

In this section literature on existing bin handling and comparable systems is reviewed. Next, the methods used in this research are described.

2.1 Literature

Before creating a model or designing scenarios, a literature study is done into existing design strategies and system definitions. Bin handling systems are a subgroup of material handling systems which only handle standardized bins, Furmans et al., 2010. Since production environments vary greatly, bin handling systems can as well. To define a bin handling system, Kay, 2012 has identified five different types of equipment, the first of which is transport equipment. This is equipment used for transportation of goods within the system from one location to another. The second is positioning equipment. Positioning equipment handles products at a single location in order to position products for further steps, often located in front of production cells. Next is unit load formation equipment, equipment used to secure materials and preserve their integrity during transport and storage, **popiela2014optimization**. In bin handling systems these are often the moulds in the bins, keeping the product in place in the bin. Fourth is the storage equipment. Equipment capable of storage for short, or sometimes longer, periods of time. Storage equipment is mainly used to cope with supply fluctuations or to store parts needed for production near the production environment. Lastly, there is the identification and control equipment. This equipment is utilized for gathering and transmitting information, crucial for coordinating the movement of materials within a facility. While Kay, 2012 focuses on traditional material handling system equipment, newer autonomous systems aim to consider functions and how they can be combined in a broader range of applications. These systems however are novel and are few and far between. Some papers such as Ye et al., 2018 talk about the use of AGVs for product and bin handling, but not in a production environment setting. However, Ye et al., 2018 does explain the combining of different functions in one unit namely the transport and positioning function.

The storage stage of a bin handling system can be further defined. In a company, goods are mostly stored in two locations: in the warehouse or in a buffer before a production cell or follow-up step, Reyes et al., 2019. The optimization of storage in warehouses is a well-practiced area of study, Karásek, 2013, and could be useful in the design of the storage stage of a bin handling system. De Koster et al., 2007 states there are two main methods of assigning storage locations: dedicated and shared storage. Warehouses are not always full. To not have empty locations in between products, and thus increasing travel distance when order picking, shared location strategies are adopted, as done by Park, 1987. Shared locations introduce the ability to use preferable locations as soon as they are free. Also, storage capacity can be based on the total demand of the system, and not on each product. Yang et al., 2017 Describes a variant of dedicated storage, a method where one location only holds one type of good, even if it is out of stock. Yang et al., 2017 describes a location assignments method called full turnover which takes into account the turn over of a product and puts the more in-demand products in the front of the ware house. Yu and De Koster, 2013 states full turnover reduces some downsides of dedicated storage, but not all: there will still be empty locations in the front.

The term 'buffer' is defined by Gupta and Jain, 2013 as a safety margin to not disrupt a process when components fail and ensure continuity. In supply chain systems a buffer is often a small, local stock of items to accommodate fluctuations in supply rate. Mahadevan and Narendran, 1993 identifies a second type of buffer, a central buffer, as shown in figure 1. The type of buffer in a bin handling system is subject to different requirements then a warehouse, Umirzoqov, 2020. In production environments, all demand will always be met. Starting production with one part of the product not in stock will hold up production immediately. The number and quantity of products is known in advance, and will be constant for each shift. This makes it possible to perfectly tune the size of the buffer capacities of the bin handling system. Problems emerge when one production line can produce multiple different products, Mahadevan and Narendran, 1993.

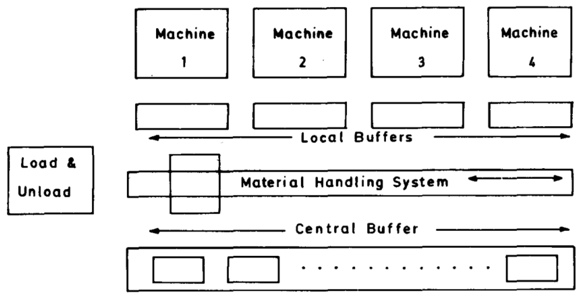


Figure 5. Layout of the system considered for the current study

Figure 1: Schematic view of buffer types and their locations, by Mahadevan and Narendran, 1993

2.2 Methods

To be able to compare results of different design strategies, the best possible solution of each system should be determined. In this section the methods to define the best possible solution are explained.

2.2.1 Mathematical modeling

Implementation of a BHS can be done in an optimal way. The goal of a BHS is to support production. Product takt time is dictated by the production line, and the BHS is supposed to be able to handle it. Therefore the demand of products is dictated by the production line and the BHS should be designed accordingly. In order to ensure the BHS can cope with all demand made by the production line, a linear deterministic mathematical model is constructed. Gong and de Koster, 2011 talks about uncertainty sources in warehouses and production environments and states only unpredicted unit failure introduces significant uncertainty in such systems. Also, deterministic models are inherently less computationally heavy the stochastic models as stated by Kenton and James, 2021. Therefore a deterministic model is preferred. This model contains equations describing the relations between different system units, demand and capabilities. To meet demand, different choices can be made. By integrating the model of the BHS into a computational solver the best possible solution can be found. For this research the solver Gurobi Optimizer is used.

The model will include four different cost sources. First are the bill of material costs. These include all the costs related to the unit parts. Second are the assembly costs. Since units arrive as parts, they need to be assembled. This is the job of a mechanic, who is paid wages. These costs are defined as the time it takes one full-time equivalent (FTE) to assemble all units combined with the costs of an FTE. Thirdly as the Installation costs. Similar to the assembling costs, mechanics are needed in this step and need to be paid. Lastly, the floorspace costs are included in the model. Floorspace costs are required since floorspace has a depreciation time. This cost source is the only one which is time related. The time on which these costs are based is the life time of the units, so the model will determine the total lifetime costs of the system.

The model not only finds the optimal solution, it also generates a design of the production environment for the proposed solution. In this design it can be directly checked on how the model makes its choices and where units should be placed.

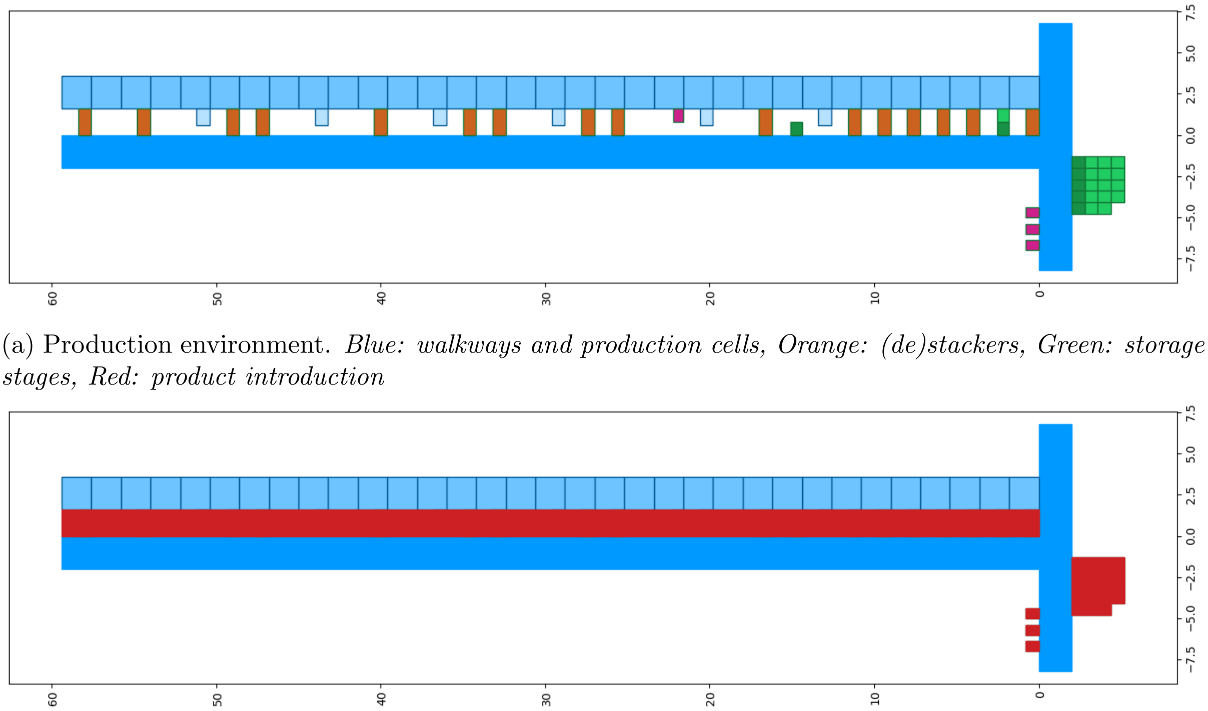
2.2.2 KPIs

The term optimal solution should be further defined since this solution depends on the optimization direction. In this research three different optimal solutions are generated.

Bill of material. The first of which is the bill of material (BOM) of the system. Each unit in the system had its own BOM. By optimizing for the total BOM, the model will prioritize less expensive units. This could also mean unit placement will increase system footprint, as long as the BOM costs decrease.

System footprint. The second KPI is the total system footprint. Floorspace can be very costly, especially in the production environment. Having a system with a small footprint means more production facilities can be placed in the building. Footprint optimal solutions can increase the BOM in order to place units in more strategic locations. Also, more expensive units could be used in order to reduce footprint. System footprint should be defined the same way for all system designs. To that purpose a definition is proposed in this paper which not only includes unit footprint but spaces between them as well. Placement of units in certain locations could restrict use for other activities of areas around the units. Therefore a definition including ”wasted space” is proposed. In figure 2 an example is given of the total system footprint according to this definition.

Total system costs. Lastly, the total system costs are defined as a KPI. In the total system costs the model will find an equilibrium between BOM, assembly, installation and floorspace costs and find the overall most cost effective solution.



(a) Production environment. *Blue: walkways and production cells, Orange: (de)stackers, Green: storage stages, Red: product introduction*

(b) System footprint

Figure 2: Model optimal solution to system implementation

3 System analysis

This paper includes a case study of an existing BHS at Prodrive Technologies. This BHS is further identified in section 3.1. Next, in section 3.2 the model is constructed and implemented on the system.

3.1 System overview

Bin handling systems can consist of any number of different units. The case study BHS gives insight into how different functionalities are allocated to different robots. Also, this system gives a benchmark to which future design changes can be compared to. The production demand, line and environment will be used for future design scenarios as well. Therefore it is important to first understand this system.

The system operates in an environment with a 60 meter long, straight production line, as can be seen in figure 2. This production line consist of 33 individual steps, of which 15 are assembly steps. For these assembly steps, the cells need semi-finished products. These products arrive in bins. Since some products are small and others are large, the number of products per bin differs. This means the number of bins needed per shift differ as well, since each cell uses one semi-finished product per finished product. Thus, the takt time together with the number of products per bin dictates the demand of each individual cell. Bins are transported through the system as stacks of bins. Since there two type of bins, 50 mm and 75 mm high, stacks can contain 5 or 7 bins. Also, due to unit constraints, only dedicated storage strategies are possible in this system.

In figure 3 a render of the system is shown. Here the four distinctly different units can be seen. The first unit (1) is the single payload exchange location (SPEL). This unit is used for introduction of products into the system, but can also be used as storage for one stack of bins. The functions of these units are defined as: item introduction / extraction and storage. The second unit (2) is the main storage stage of the system called multi bin cabinet (MBC). This unit is modular and can be combined with itself to form deep rows of storage locations. To achieve this, various versions of this unit have been developed: the main MBC in the front (MBC Master), the back expansion (MBC HBT) and a back docking version (MBC Slave). The MBC Master is always the first in the row since this version had both communication and AGV docking capabilities. The MBC HBT version costs half as much as the Master version but does not include communication or docking capabilities because it does not need them when connected to a Master.

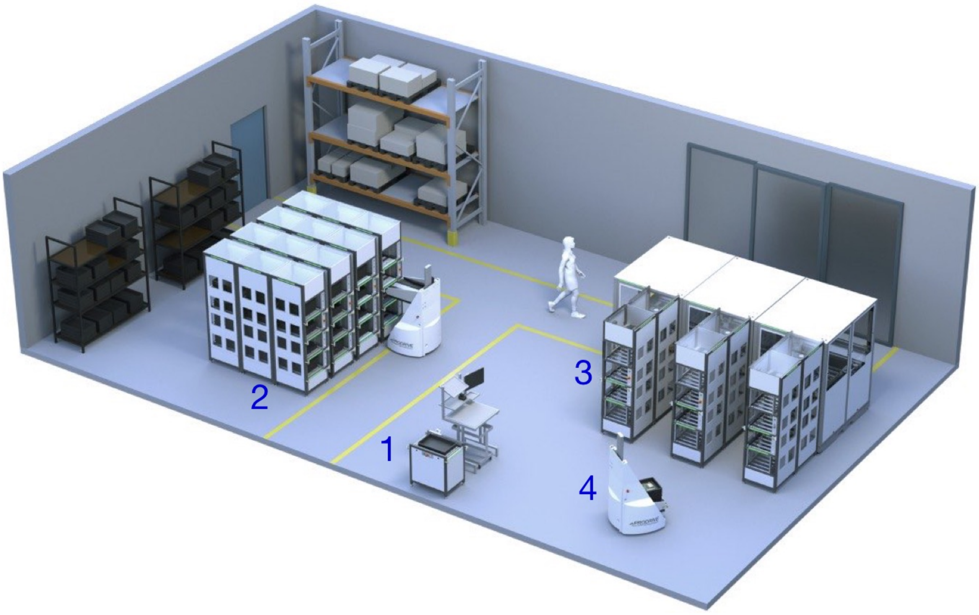


Figure 3: Render of case study bin handling system.

(1) SPEL (2) MBC (3) DSCAB (4) AGV

The MBC Slave costs slightly less than the Master version but does include AGV docking capabilities, which allow for (un)loading from the back of a row of MBCs. The main function of the MBCs is storage. The third (3) unit in the system is the (de)stacker cabinet (DSCAB). The DSCABs are designed to enable the transportation of stacks through the system by allowing the destacking of bins before reaching the production cells. The DSCABs contain two internal stack storage locations, two payload exchange locations and two (de)stacker modules. The DSCAB function are: (de)stacking, item introduction/extraction, storage. The fourth and last unit in the system (4) is the automated guided vehicle (AGV). The AGVs handle all transportation of stack, or bin, between the other units. Functionality of the AGV is: item transportation. To summarize, the functions identified from this systems are:

- Item introduction / extraction (operators)
- Item introduction / extraction (production cells)
- Storage
- Stacking
- Destacking
- Item transportation

3.2 Model construction

Next, the model for the case study BHS is constructed. This model should determine both unit count and unit placement. First unit count constraints are added. A decision variable representing a three dimensional grid of storage locations, called MBC_{grid}, is introduced. Since only MBC Masters have both connection and docking capabilities, the first unit placed should be a Master. This is stated in equation 3.1.

$$MBC_{grid}[l,m,k] \geq MBC_{grid}[l,m,(k+1)] \quad (3.1)$$

Here l is the set of vertical storage locations (height), k the set of locations behind each other (depth) and m the locations next to each other (width). Next all unique storage locations are identified. Since dedicated storage does not allow for location sharing, each $[l,m]$ location of MBC_{grid} represents a unique storage location of size k.

$$MBC_{usl}[l,m] = \sum_{k=1}^{K} MBC_{grid}[l,m,k] \quad (3.2)$$

MBC_{usl} can be constructed from the summation of MBC_{grid} since MBC_{grid} is binary, as shown in equation 3.2. To include the ”wasted space” into the model and enable costless placement of units here, equation 3.3 is added.

$$\sum_{k=1}^{K} MBC_{grid}[0,m,k] \leq MBC_{max}[m] \tag{3.3}$$

MBC_{max} is a variable indicating the locations available for MBC placement in each row m. m has a subset p indicating the "wasted space" and its locations. To determine whether the demand of each product is met, the decision variable $Assign[l,m,i]$ is introduced. Also, the parameter $B_s[i]$ is constructed which indicated the bin demand per product.

$$\sum_{l=1}^{L}\sum_{m=1}^{M}(Assign[l,m,i] * MBC_{usl}[l,m]) \geq Bs[i] \tag{3.4}$$

In equation 3.4 the relation which ensures placement of all products is depicted. From MBC_{grid}, the different MBC unit quantities can be extracted. The quantities of SPELs and DSCABs are dictated by system limitations and are directly copied from model initiation data. The number of AGVs required is determined by the equation 3.5.

$$AT = \sum_{n=1}^{N}(MBCDSCAB_n + IntroMBC_n) * ATI_n$$
$$*(\sum_{i=1}^{I}(Bs) - Bs_{-1}) + FinishMBC_n * ATI_n * Bs_{-1} \tag{3.5}$$

Here ATI_n includes data on AGV travel and action time and other statements indicate the different actions per job. From these variables, the cost functions, and thus the objective function, can be constructed. The objective function depends on the optimization objective, being footprint, BOM or total costs.

$$OBJ = \begin{cases} C_b + C_a + C_i + C_f & \text{Total costs} \\ C_b & \text{BOM costs} \\ C_f & \text{System footprint} \end{cases} . \tag{3.6}$$

4 Model implementation

Now that a model has been constructed, it is possible to test and evaluate various design scenarios. This section explains eight different scenarios and presents their results.

4.1 Scenario design

With the case study as a reference, changes are made to the system introducing different buffer location usage, storage strategies and function combinations. **1) Case study.** First the results of the case study BHS are evaluated. **2) Usage of the internal storage locations of the (de)stackers.** The DSCABs have a small local buffer, as explained by Mahadevan and Narendran, 1993. Usage of these buffers will decrease the storage capacity needed by the central buffer. Since these storage locations are already present, no unit modifications are needed. **3) Expanded local buffers in the (de)stackers.** By increasing the size of the local buffers, the central buffer size can be even smaller. To do this, a unit modification is required. This is assumed to increase DSCAB unit BOM by 30%. **4) Shared storage strategy.** The case study BHS is bound to dedicated storage strategies. In this scenario, different products can be placed in the same USL, reducing the need for more USLs and empty locations in the storage units. **5) Shared storage, no environment limitations.** In scenario 4, limitations on unit placement were still considered. This meant no rows with more than 4 MBCs could be constructed. In this scenario this limitation is removed and the model is free to construct rows as long as it deems fit. **6) Centralized (de)stacking.** DSCABs are currently placed before each cell, even though some cells have such a small demand the DSCAB is idle for 99% of the shift. In this scenario an central (de)stacker is used and AGVs will transport bins instead of stacks. **7) Mobile (de)stackers.** As mentioned above, some DSCABs are idle most of the time. In this scenario the DSCABs are outfitted with a unidirectional movement module. They can travel along the production line, serving multiple cells. This means a unit modification is required, and thus an increase in BOM. This increase is estimated at 100%. **8) Mobile (de)stackers - no obstacles.** The unidirectional movement of the DSCABs in scenario 7 means the DSCABs can not pass obstacles along the production line. By relocating obstacles, the DSCABs can now travel along the entire production line. **9) AGV with (de)stacker modules.** Taking scenario 8 one step further, AGV now have the ability of both (de)stacking and omnidirectional movement. This integrates the functions: item transportation and (de)stacking. Since the AGVs now have to be outfitted with a (de)stacker module, the BOM is estimated to increase with 50 %.

4.2 Results

Takt time - 60s	Scenario								
	1	2	3	4	5	6	7	8	9
Total costs	100%	96%	106%	100%	100%	85%	88%	76%	74%
BOM costs	100%	95%	110%	100%	100%	67%	82%	51%	64%
Floorspace	100%	99%	99%	100%	100%	136%	106%	151%	101%
Floorspace m2	99,0	98,4	98,4	99,0	99,0	134,8	104,6	149,1	99,5

Table 1: Summary of results w.r.t case study - floorspace optimal solutions

Takt time - 60s	Scenario								
	1	2	3	4	5	6	7	8	9
Total costs	100%	96%	105%	99%	94%	85%	88%	76%	74%
BOM costs	100%	96%	111%	99%	92%	69%	83%	52%	63%
Floorspace	100%	98%	96%	100%	98%	122%	103%	135%	100%
Floorspace m2	110,2	108,0	105,7	110,2	108,0	134,8	113,0	149,1	110,7

Table 2: Summary of results w.r.t. case study - BOM optimal solutions

In tables 1 and 2 results from both BOM and Footprint optimal solutions are depicted. In appendix B and C the entirety of the results are depicted, both quantitative results and generated environment designs.

5 Conclusion

This thesis aims to inform the reader on the performance of different design choices of autonomous bin handling systems. In section 5.1 the findings of this research are summarized. Lastly, in section 5.2 recommendations on future work are given.

5.1 Conclusions

Autonomous, flexible bin handling systems are state of the art. Designing such systems therefore is often done with few examples or knowledge of performance alteration design strategies. This thesis composed a quantitative optimization model which is used on different design strategies of bin handling systems.This is done in order to find an answer on the main research question: **what are the effects of different design strategies of an autonomous bin handling system on costs, floorspace and throughput capabilities?** Design strategies researched in this thesis are compared to the case study system. Conclusions are drawn in reference to performance of the optimal solutions for the case study system

Firstly, with the definition of system footprint this thesis proposes, only small reductions of 2 to 4 % in footprint are found. Reductions found are mainly caused by the placement of units in bill of material wise undesirable locations. A second factor found for the reduction of systems footprint is the extended use of local buffers. Local buffers are located in the already existing equipment and thus do not increase system footprint. In contrast to warehousing, shared storage strategies have minimal impact on both costs and floorspace in bin handling systems. Design choices such as mobile (de)stacker equipment and bin transportation instead of stack transportation proved to have a negative impact of floorspace. Mobile (de)stackers require the relocation of obstacles, or other equipment, to reach their full potential. This increases the system footprint drastically by 35 %. The transportation of bin instead of stacks introduced more traffic in the environment. This traffic leads to the need for wider walkways and thus an increase in footprint. This increase can amount to up to 22 %.

Secondly, design strategies are evaluated for their bill of material costs. While extended local buffers have a positive influence on floorspace, the system BOM increases by 10 %. The individual BOM of existing units increases due to the requirement a larger internal storage location, which ultimately increases the BOM. Shared storage strategies, which are common in warehouse optimizations, are also tested. Due to limitations on available space in the production environment there is only a 1% improvement in BOM. When removing space limitations, this improvement grows up to 8 %. While increasing system footprint, bin transportation instead of stack transportation decreases the BOM by 30 %. This seems counterintuitive since transporting bin would seem to lead to the need for an excessive amount of transporting units. This reduction is due to the elimination of most (de)stacker units, and a quantitative optimal solution for resupply planning. The largest savings in BOM are found when adding mobility functions to (de)stacker equipment. Here a decrease of 48% can reached. When fully integrating the transportation and (de)stacking function in one unit, a positive result for the system BOM is found. This strategy does not seem to have a negative result for the system footprint. A reduction of 36 % is obtained when combining these function in one unit, while keeping system footprint constant with the case study.

Lastly, the throughput capabilities of the design strategies are evaluated. Product takt time is decreased and the changes in results are compared. Noteworthy is the the improved result for shared storage without placement limitations. Here the total system costs reduced further from 6% at a takt time of 60 seconds to 9% at a takt time of 30 seconds. Effectiveness of solutions as bin transport instead of stack transport and mobile (de)stackers reduces as takt times get lower. Bin transport strategies see the total cost reduction of 15 % at a takt time of 60 seconds go to 9% reduction at a takt time of 30 seconds. Mobile (de)stackers have their total cost reduction reduce to 15 % at a takt time of 30 seconds. For this strategy, total cost reduction was 25 % at a takt time 60 seconds. Full integration of the transportation and (de)stacking function shows only a 3% less effective total cost at a 30 seconds takt time. Floorspace still does not increase in this strategy.

5.2 Future work

This thesis only touches the hardware design part of bin handling systems. Bin handling systems are part of a larger system, the production environment. In order to keep results reliable, other factors are kept constant since their influence could not reliably be estimated and quantified. For future research promising scenarios from this thesis might be used and other factors can be altered such as:

Production line layout. Placement of bin handling units is heavily restricted in this thesis due to the predefined production line placement. The production line is defined as a line. However, production lines come in all kinds of shapes and configurations. Testing the validity of scenarios given in this thesis for different production line configurations will broaden the understanding of certain design choices.

Not only the predefined production line layout but also the production environment layout limits the placement of bin handling units. Future work might consider total freedom in unit placement by having no restrictions on production environments.

This study is based on a case study. This case study has such a throughput that the employed AGVs are able to cope with demand easily. Therefore unit placement optimization for AGV performance is left outside of the scope. For future work, when assessing higher throughput systems this should be taken into account in order to find the absolute minimal number of AGVs required.

Lastly, in this thesis bin type, shape and size are based on the bins already existing in the case study. Bins play a big role in the performance of bin handling systems. Further research could look into the alteration of the bins in order to increase performance. Since bins are supposed to be universal, meaning all products should fit in them and all units should be able to handle them, all factors surrounding the bins should be considered. This thesis considers two types of bins. Future works could also consider more, or less types.

References

Baral, A. (2021). The 'design to cost' perspective.

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

Furmans, K., Schonung, F., & Gue, K. R. (2010). Plug-and-work material handling systems.

Gong, Y., & de Koster, R. (2011). A review on stochastic models and analysis of warehouse operations. *Logistics Research*, *3*(4), 191–205.

Gupta, S., & Jain, S. K. (2013). A literature review of lean manufacturing. *International Journal of Management Science* *and Engineering Management*, *8*(4), 241–249.

Jayal, A., Badurdeen, F., Dillon Jr, O., & Jawahir, I. (2010). Sustainable manufacturing: Modeling and optimization challenges at the product, process and system levels. *CIRP Journal of Manufacturing Science and Technology*, *2*(3), 144–152.

Karásek, J. (2013). An overview of warehouse optimization. *International journal of advances in telecommunications, electrotechnics, signals and systems*, *2*(3), 111–117.

Kay, M. G. (2012). Material handling equipment. *Fitts Dept. of Industrial and Systems Engineering North Carolina State University*, *65*.

Kenton, W., & James, M. (2021). Stochastic modeling.

Kroemer, K. H. (2017). *Ergonomic design for material handling systems*. CRC Press.

Mahadevan, B., & Narendran, T. (1993). Buffer levels and choice of material handling device in flexible manufacturing systems. *European journal of operational research*, *69*(2), 166–176.

Park, B.-C. (1987). Closest open location rule in as/rs. *Journal of Korean Institute of Industrial Engineers*, *13*(2), 87–95.

Reyes, J., Solano-Charris, E., & Montoya-Torres, J. (2019). The storage location assignment problem: A literature review. *International Journal of Industrial Engineering Computations*, *10*(2), 199–224.

Umirzoqov, A. (2020). Using intermediate buffer temporary warehouses inside the working area of the gold mining quarry. *Scienceweb academic papers collection*.

Wang, J., Zhang, N., & He, Q. (2009). Application of automated warehouse logistics in manufacturing industry. *2009 ISECS International colloquium on computing, communication, control, and management*, *4*, 217–220.

Yang, P., Yang, K., Qi, M., Miao, L., & Ye, B. (2017). Designing the optimal multi-deep as/rs storage rack under full turnover-based storage policy based on non-approximate speed model of s/r machine. *Transportation Research Part E: Logistics and Transportation Review*, *104*, 113–130.

Ye, Y., He, L., Wang, Z., Jones, D., Hollinger, G. A., Taylor, M. E., & Zhang, Q. (2018). Orchard manoeuvring strategy for a robotic bin-handling machine. *biosystems engineering*, *169*, 85–103.

Yu, Y., & De Koster, R. B. (2013). On the suboptimality of full turnover-based storage. *International Journal of Production Research*, *51*(6), 1635–1647.

B

Production layout

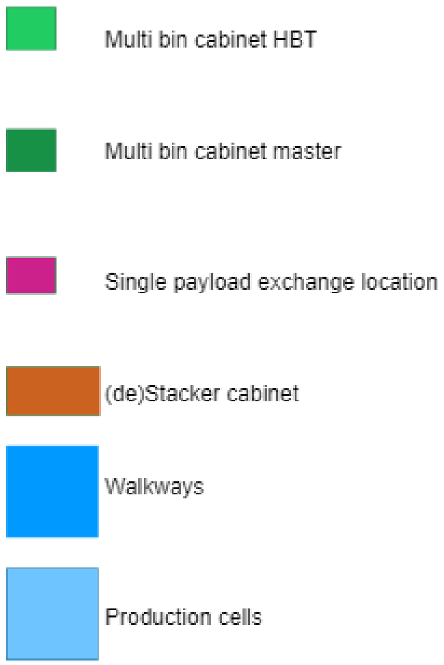
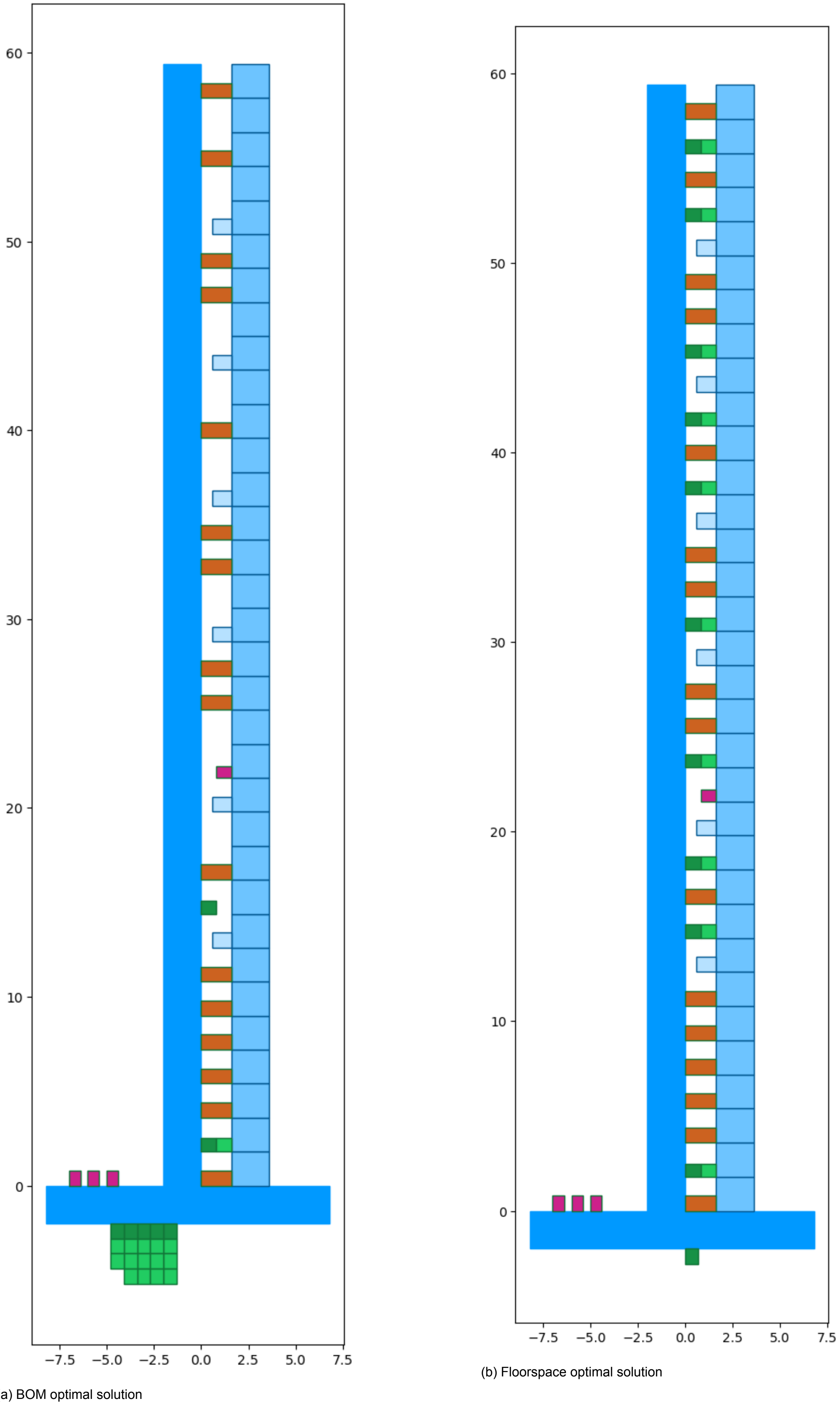


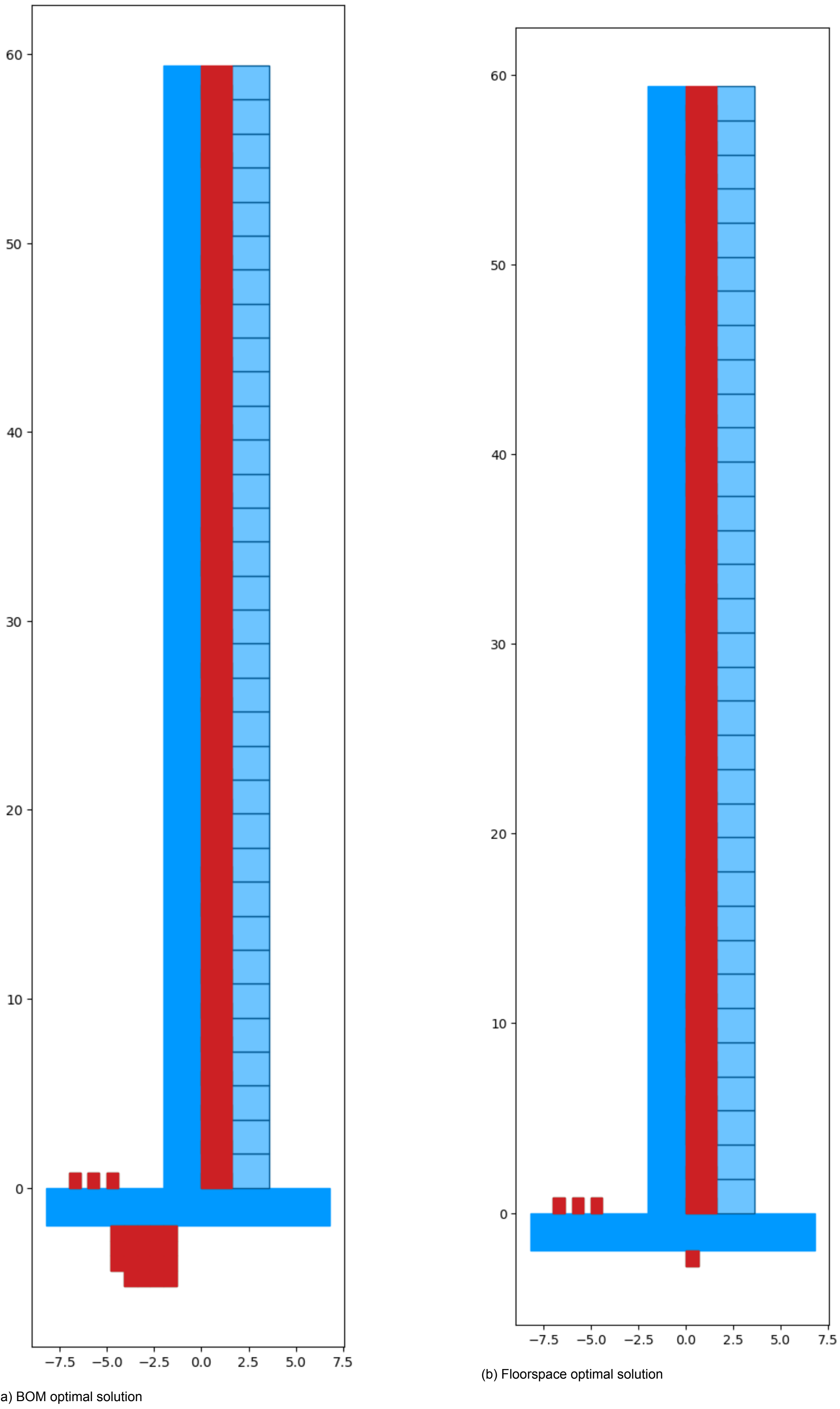
Figure B.1: Legend



(a) BOM optimal solution

(b) Floorspace optimal solution

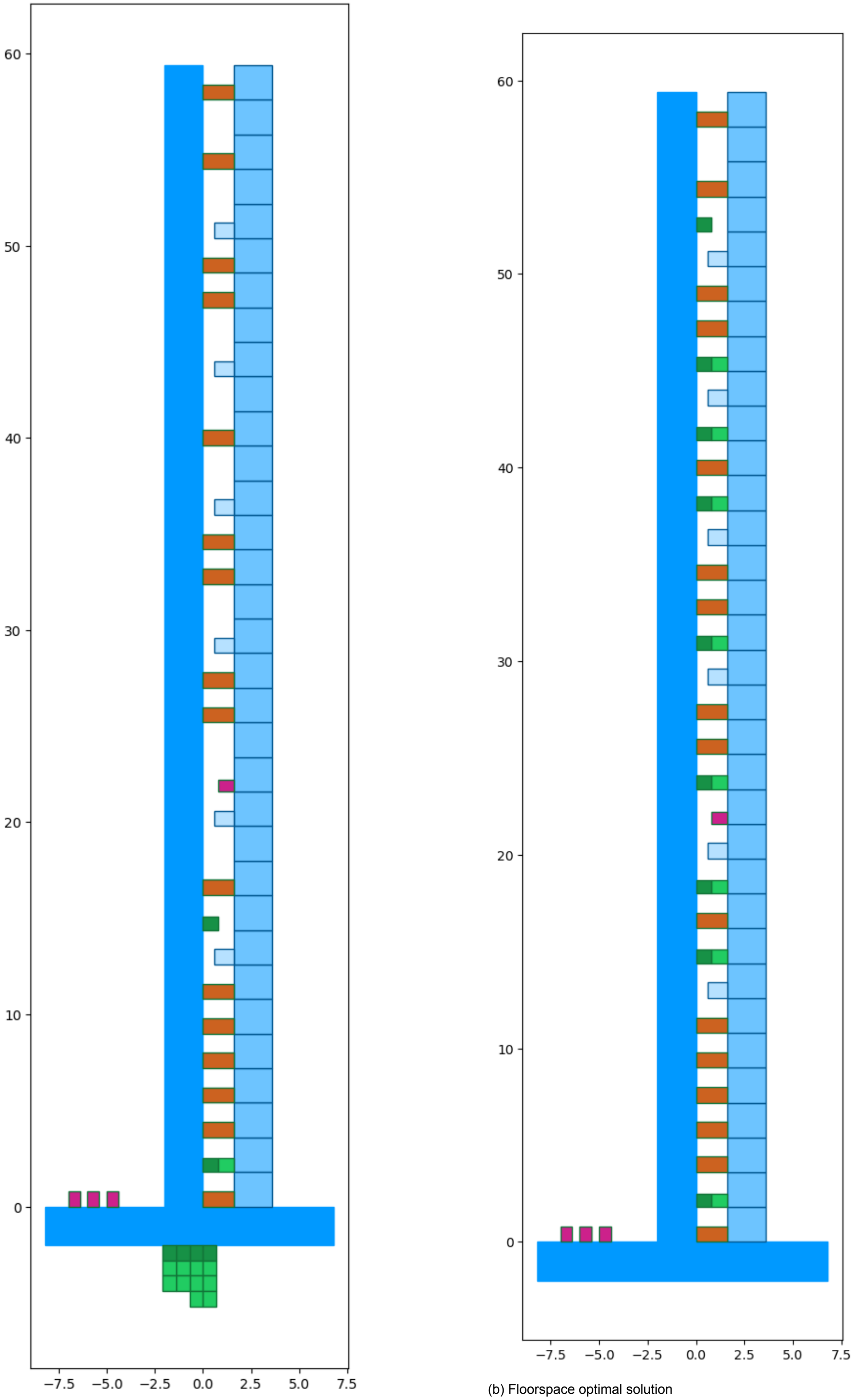
Figure B.2: Generated environment designs - case study



(a) BOM optimal solution

(b) Floorspace optimal solution

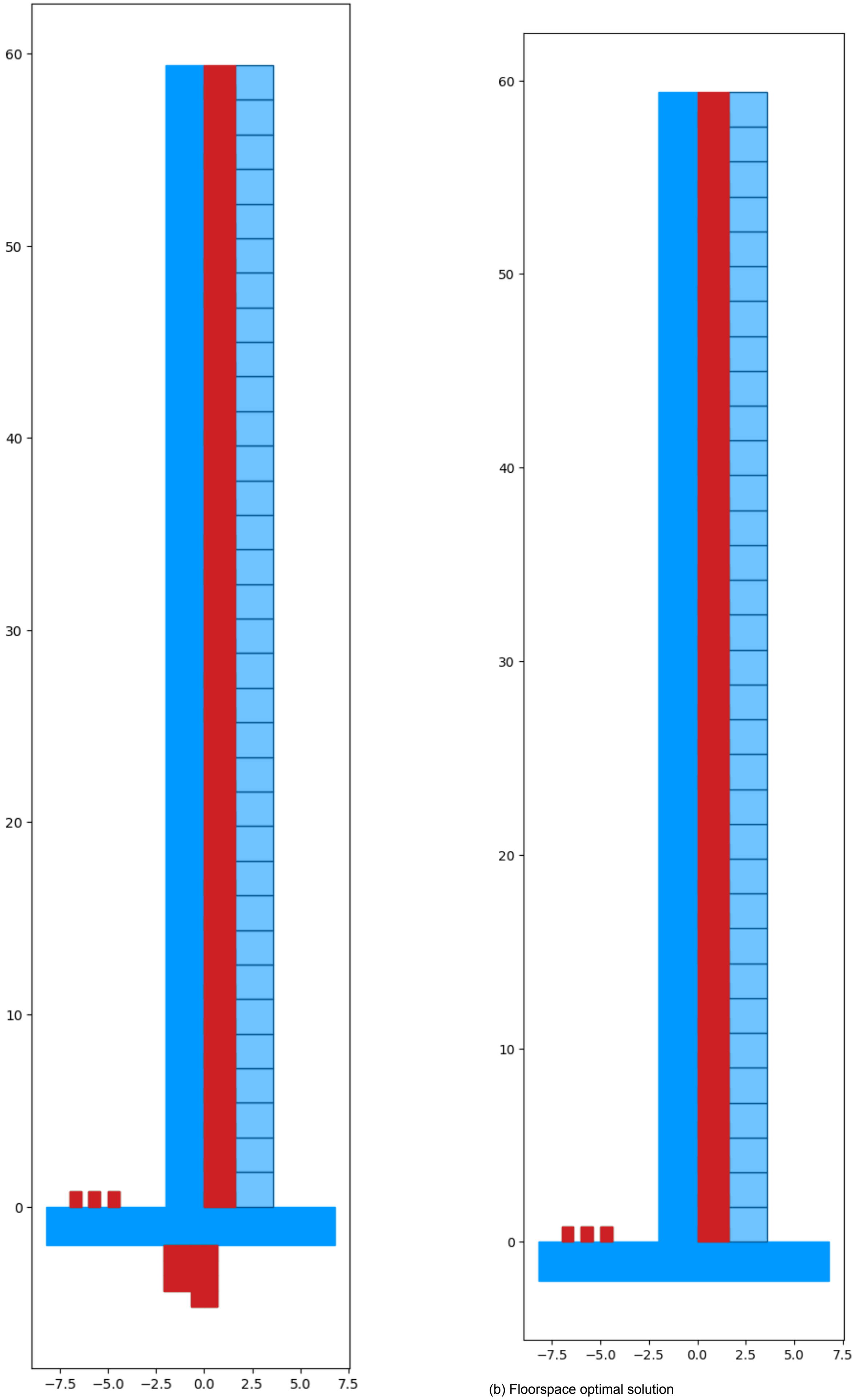
Figure B.3: System footprint of optimal solutions - case study



(a) BOM optimal solution

(b) Floorspace optimal solution

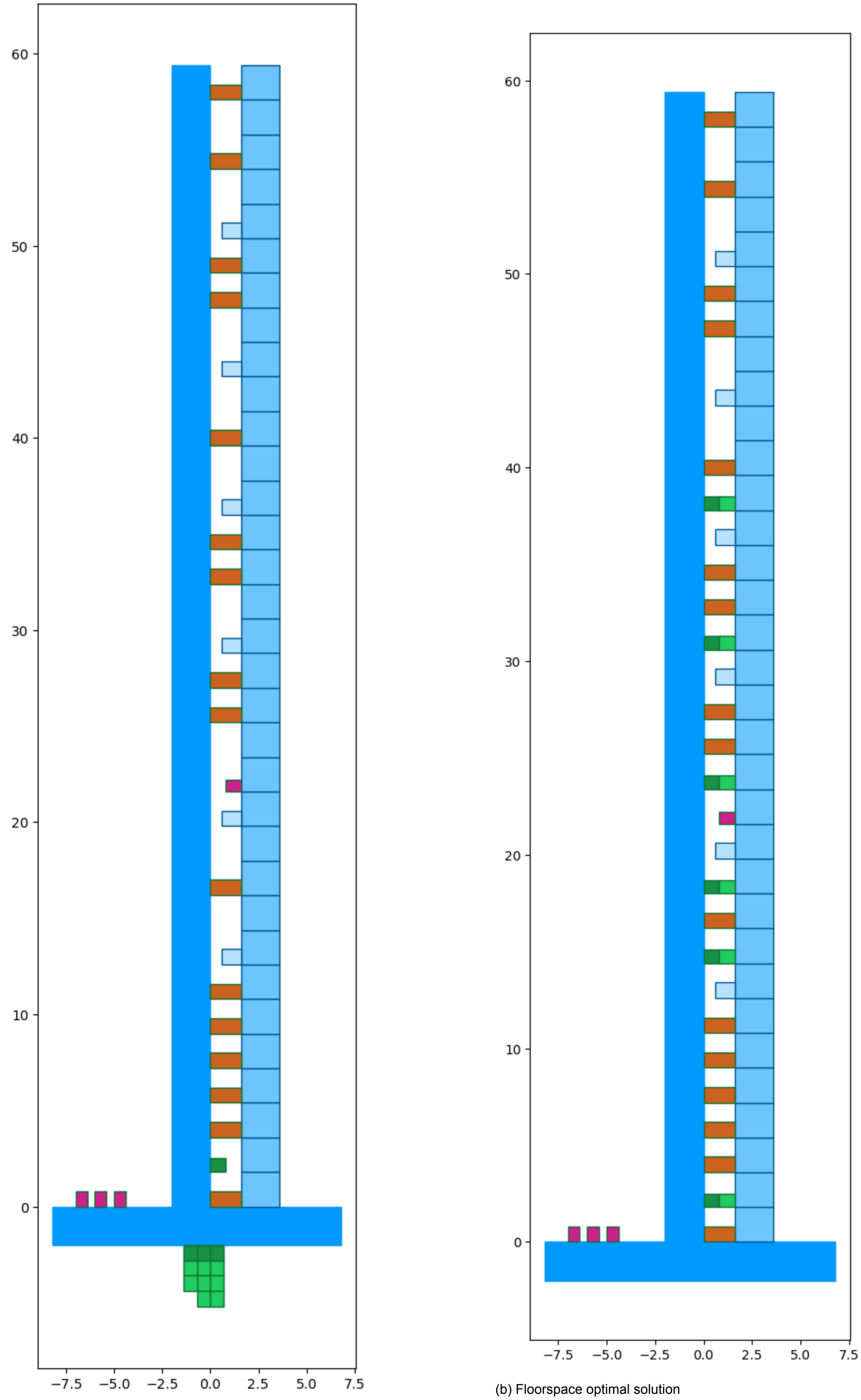
Figure B.4: Generated environment designs - experimental setup 2



(b) Floorspace optimal solution

(a) BOM optimal solution

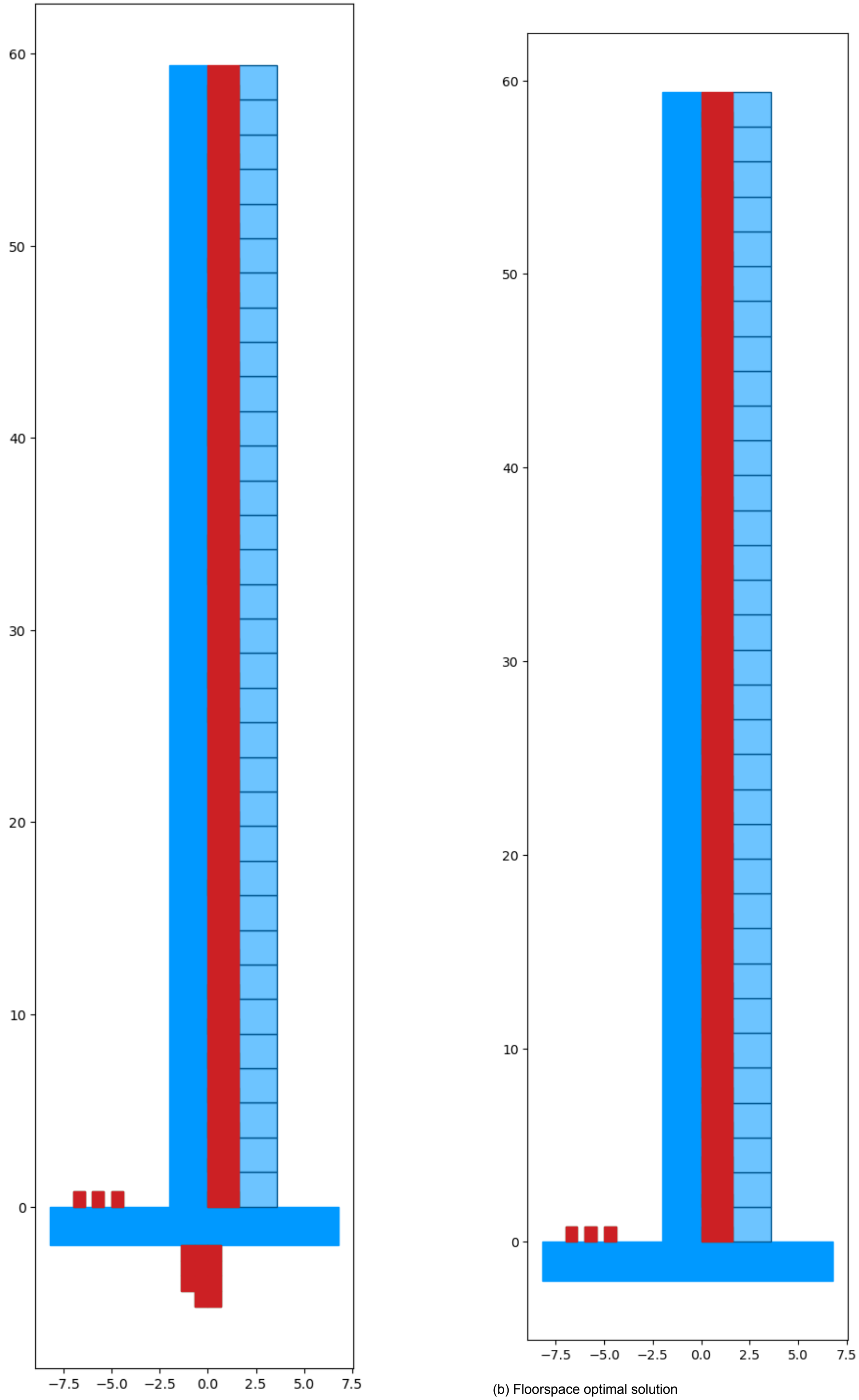
Figure B.5: System footprint of optimal solutions - experimental setup 2



(a) BOM optimal solution

(b) Floorspace optimal solution

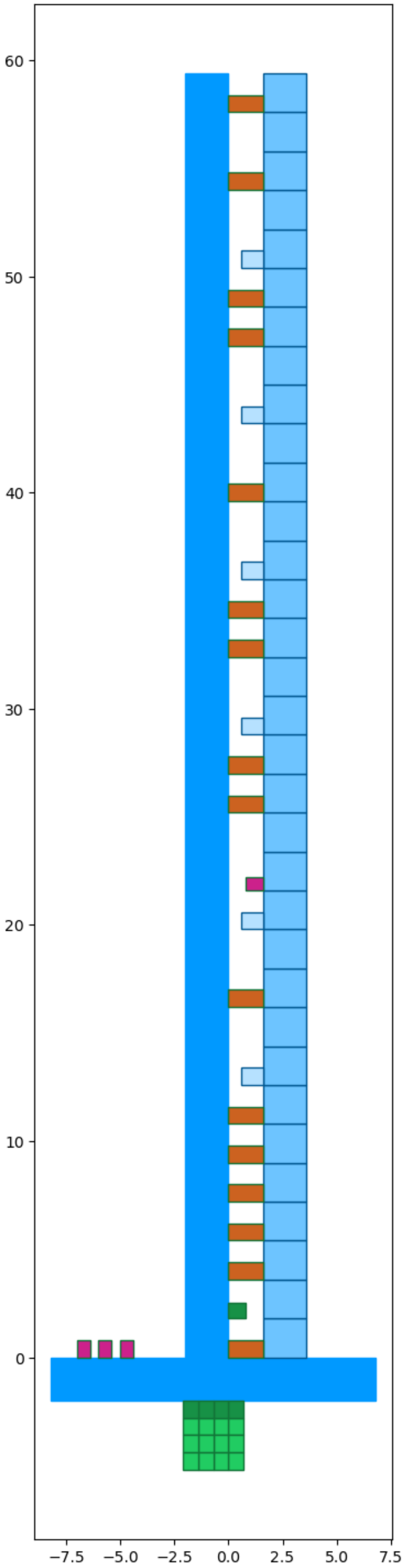
Figure B.6: Generated environment designs - experimental setup 3



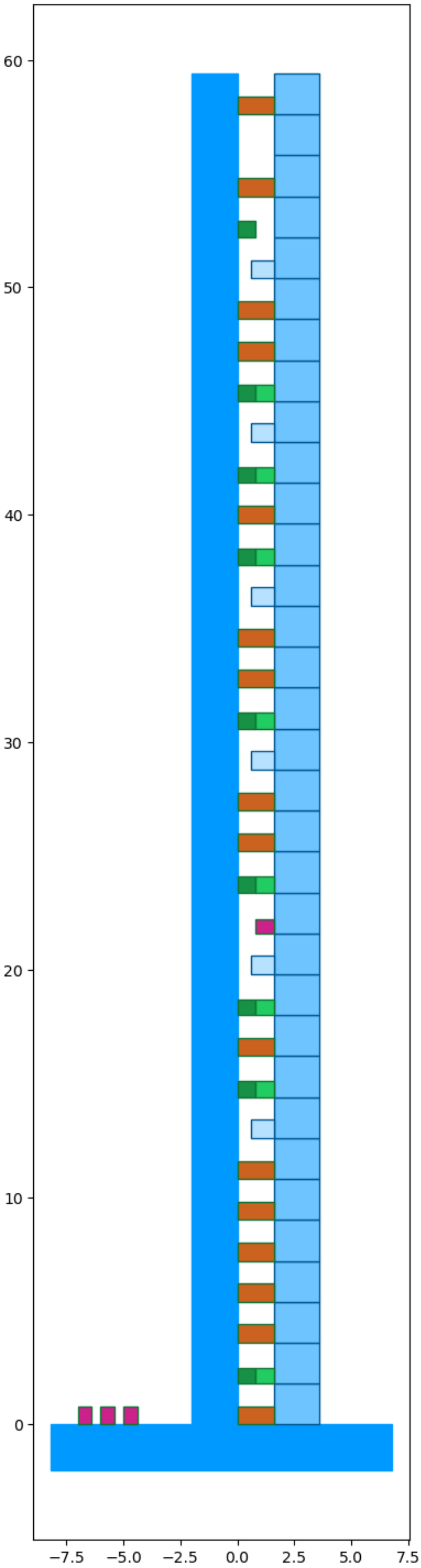
(a) BOM optimal solution

(b) Floorspace optimal solution

Figure B.7: System footprint of optimal solutions - experimental setup 3

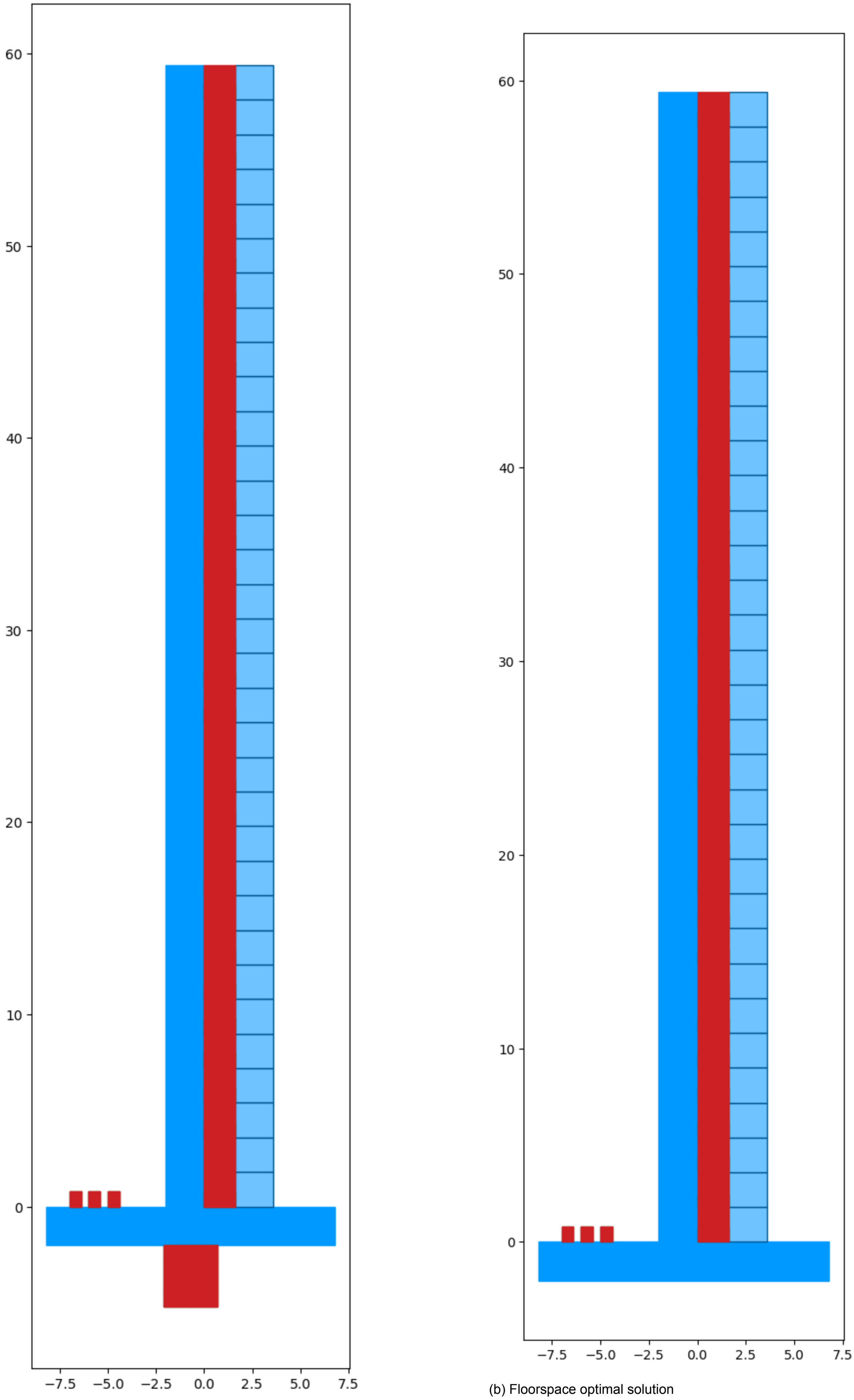


(a) BOM optimal solution



(b) Floorspace optimal solution

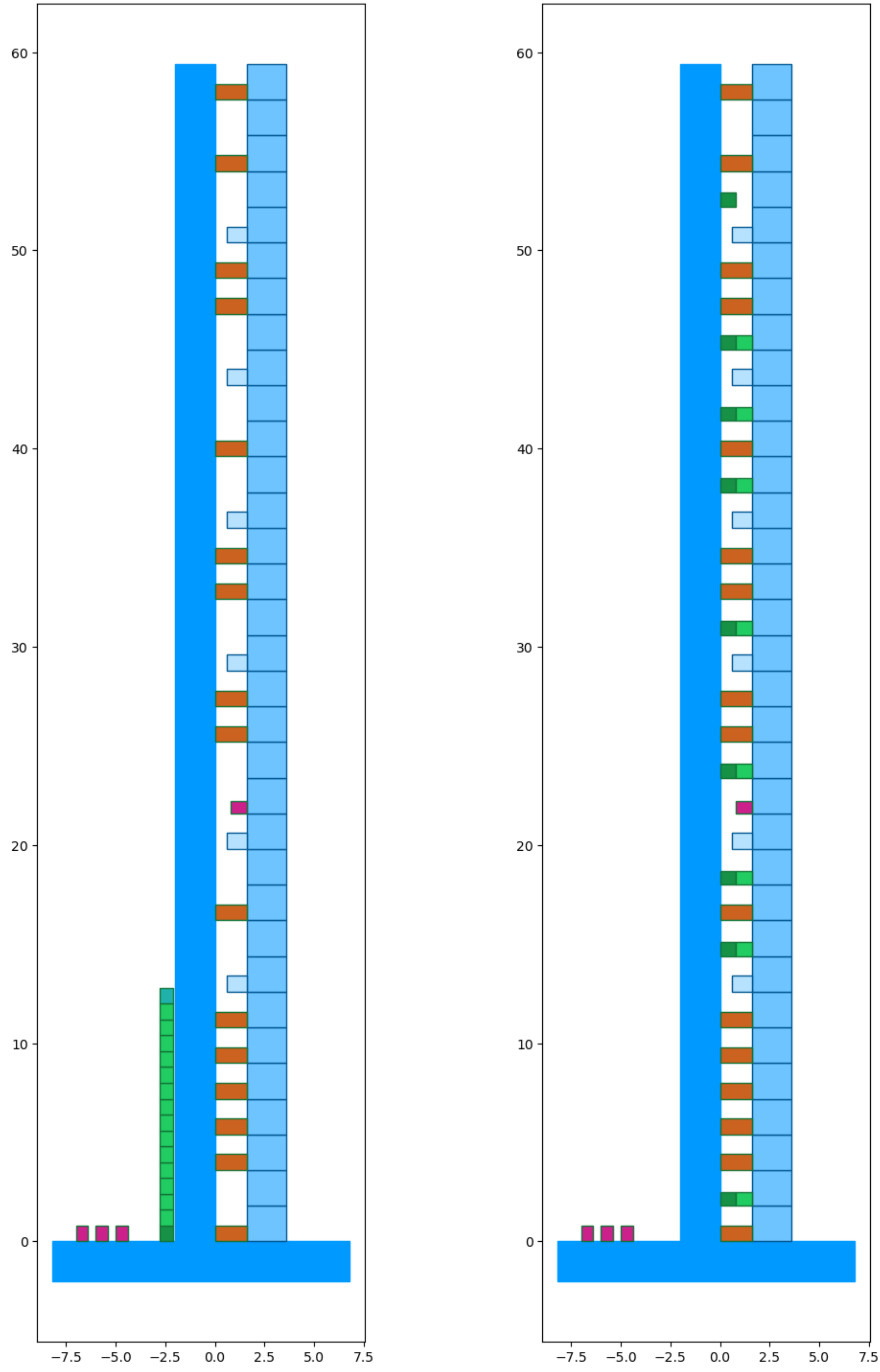
Figure B.8: Generated environment designs - experimental setup 4



(a) BOM optimal solution

(b) Floorspace optimal solution

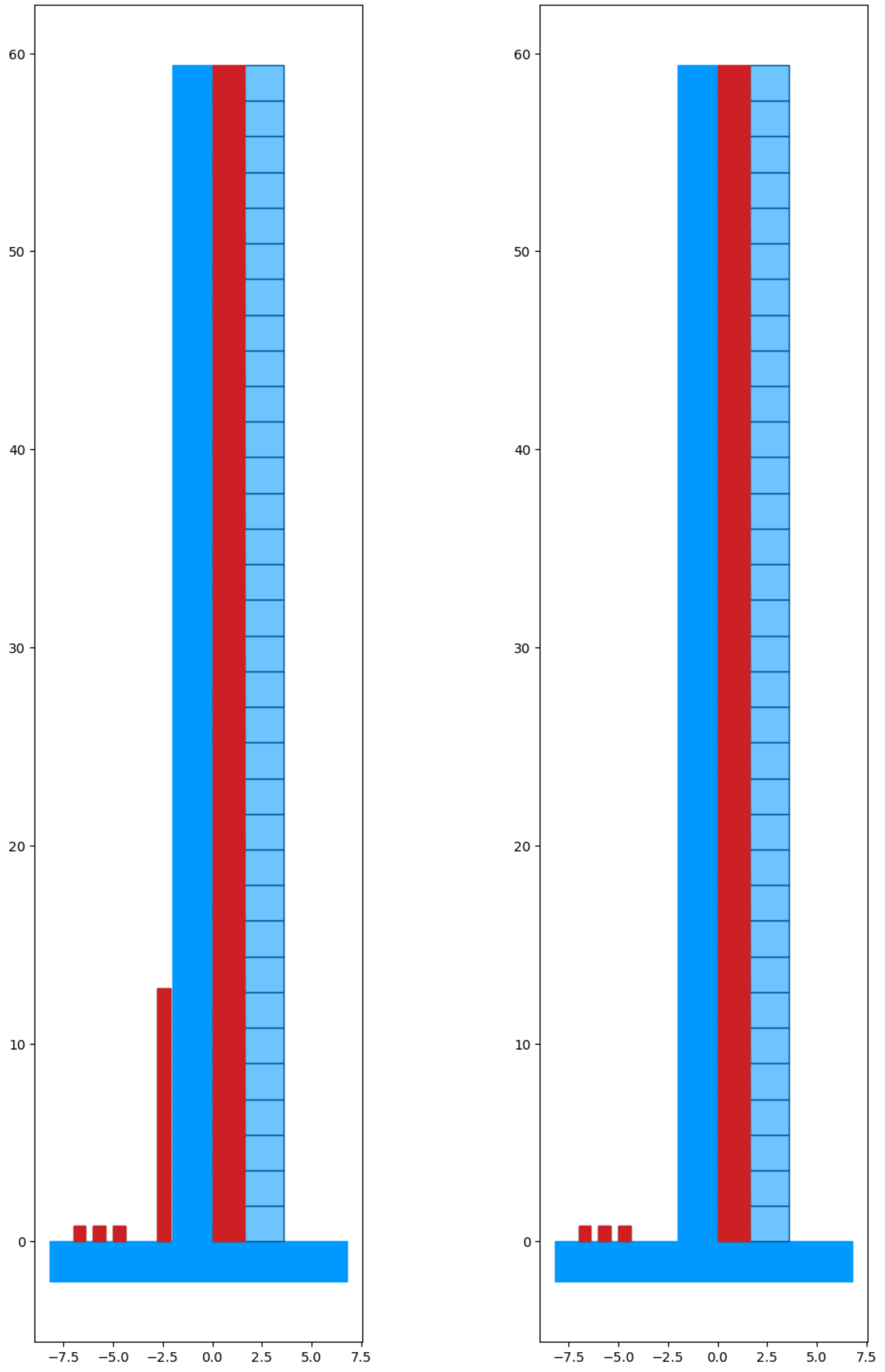
Figure B.9: System footprint of optimal solutions - experimental setup 4



(a) BOM optimal solution

(b) Floorspace optimal solution

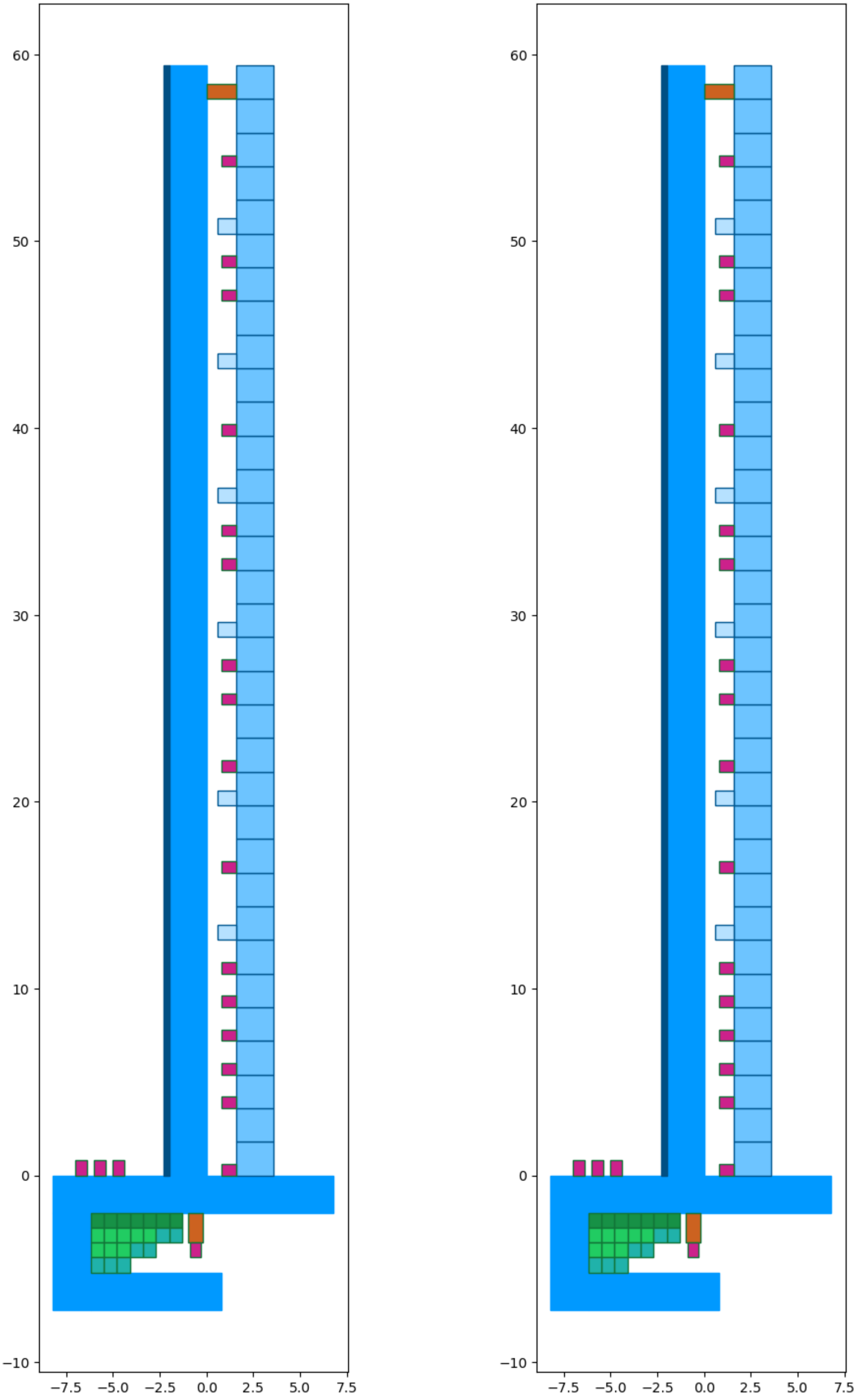
Figure B.10: Generated environment designs - experimental setup 5



(a) BOM optimal solution

(b) Floorspace optimal solution

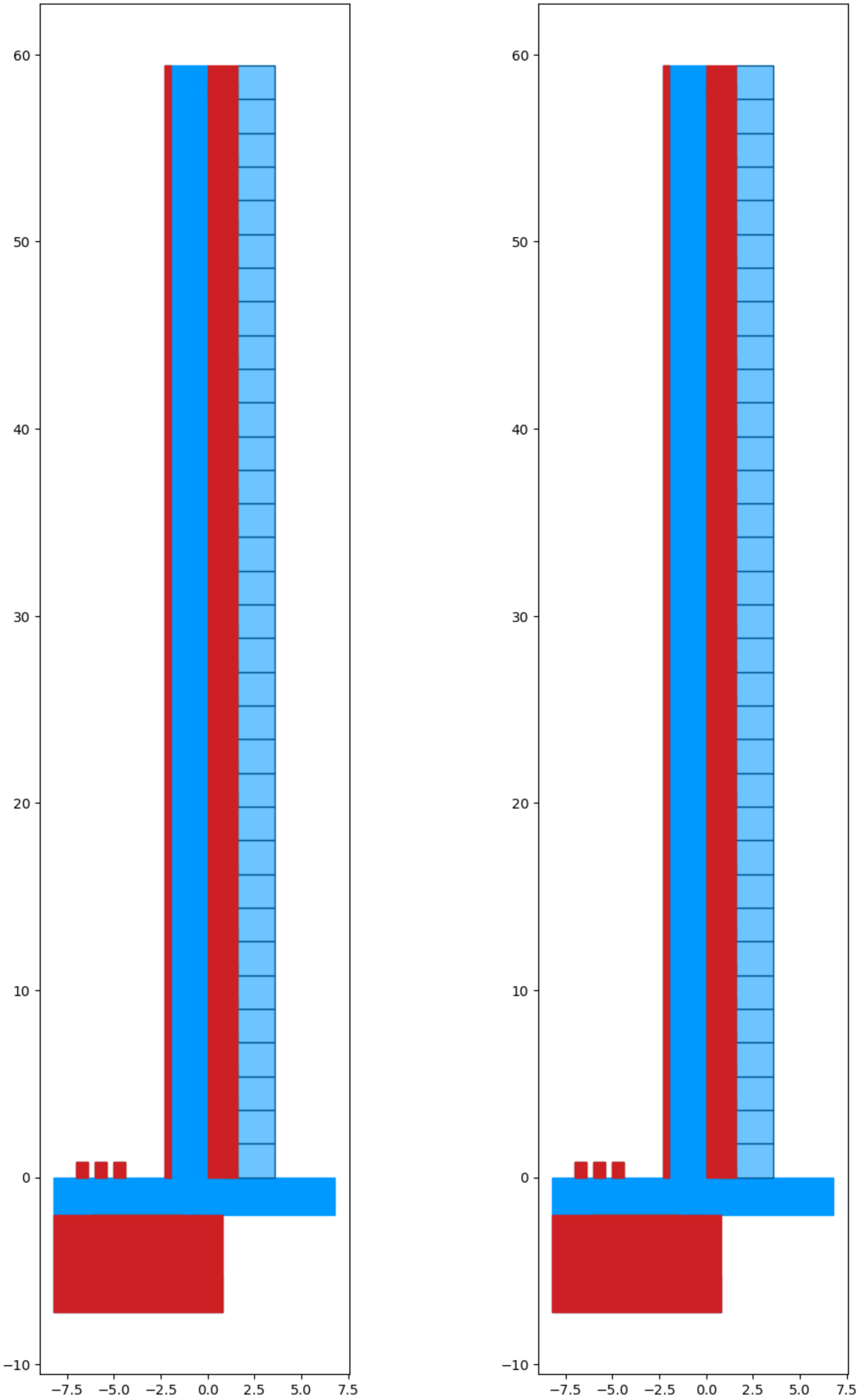
Figure B.11: System footprint of optimal solutions - experimental setup 5



(a) BOM optimal solution

(b) Floorspace optimal solution

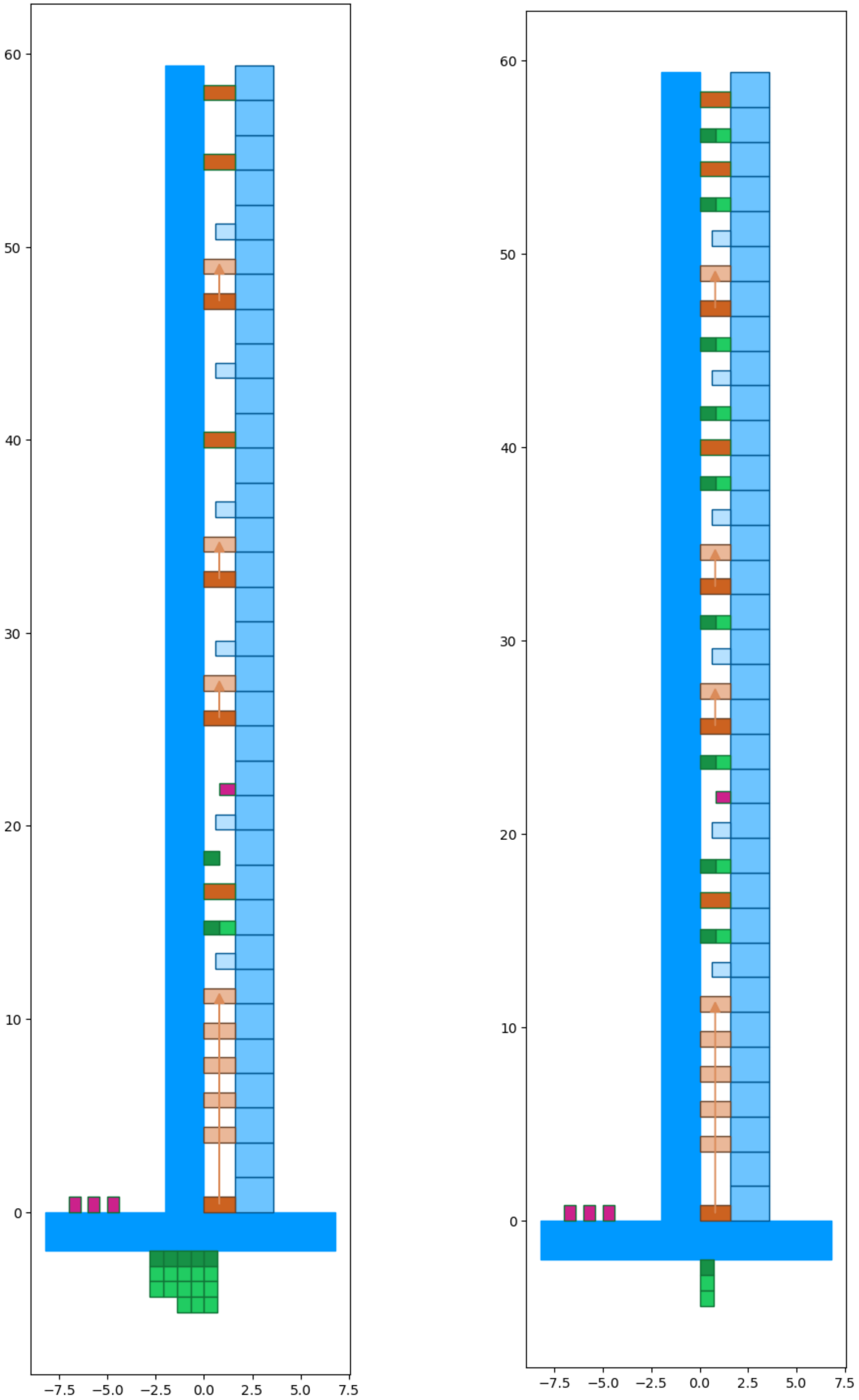
Figure B.12: Generated environment designs - experimental setup 6



(a) BOM optimal solution

(b) Floorspace optimal solution

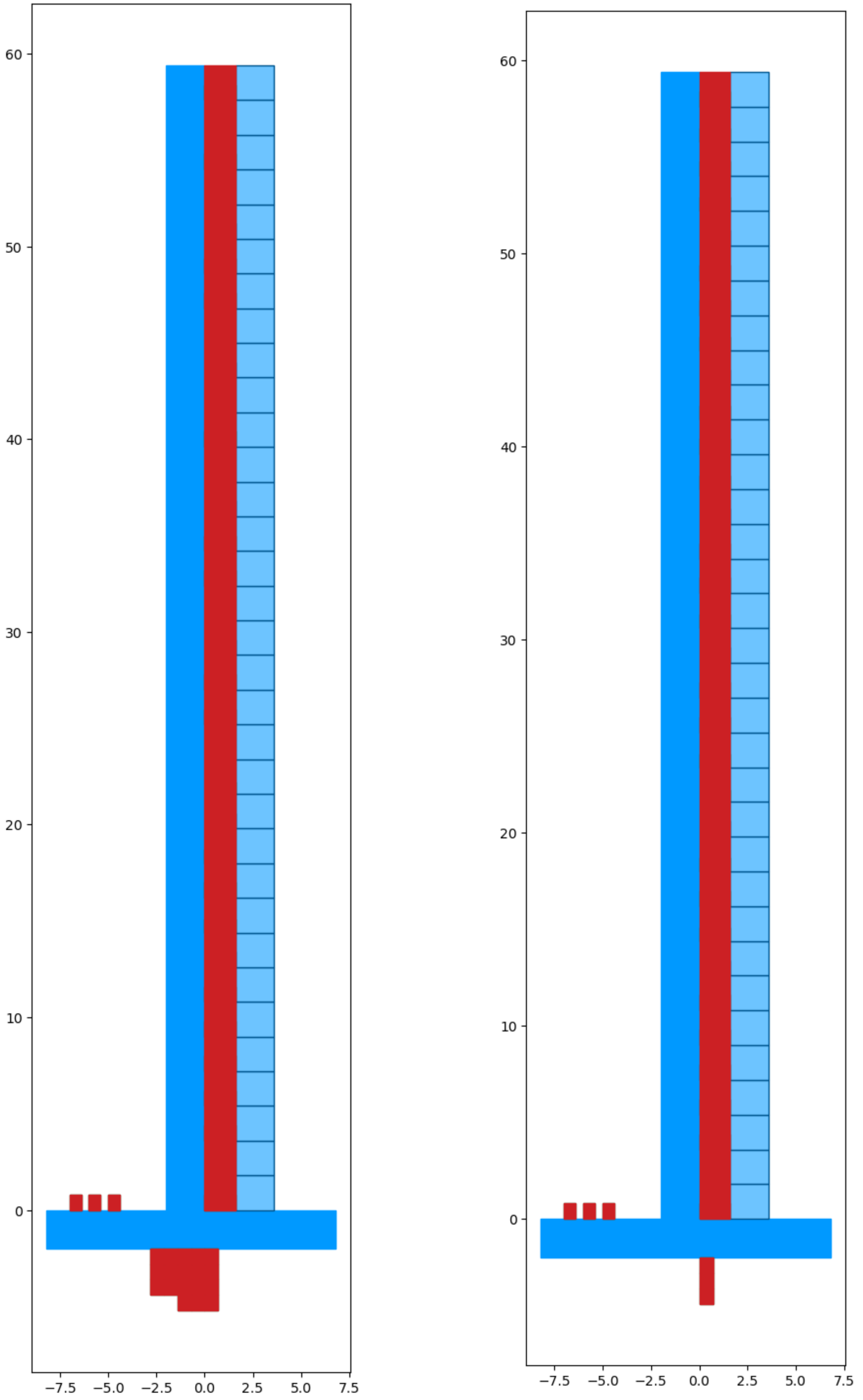
Figure B.13: System footprint of optimal solutions - experimental setup 6



(a) BOM optimal solution

(b) Floorspace optimal solution

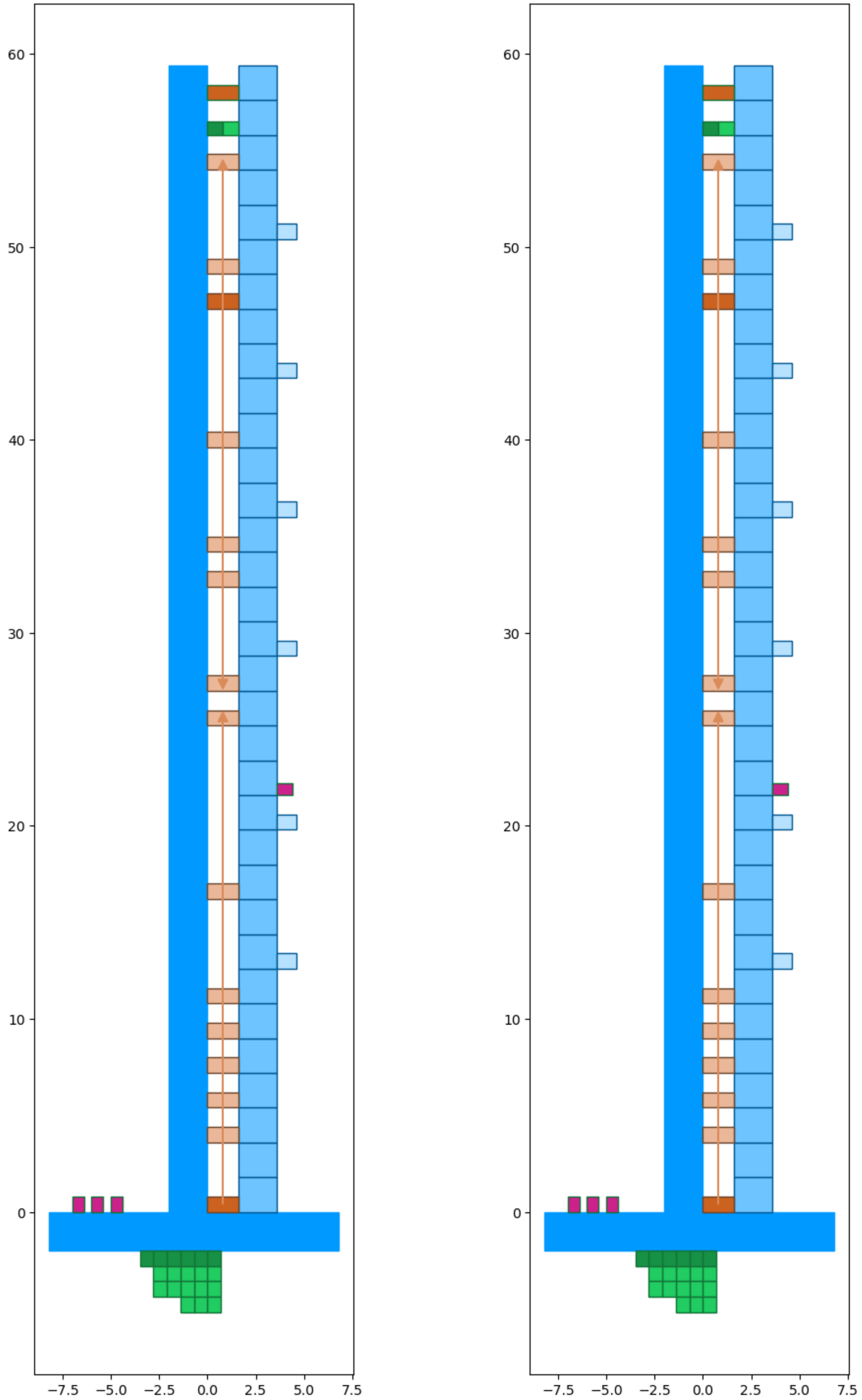
Figure B.14: Generated environment designs - experimental setup 7



(a) BOM optimal solution

(b) Floorspace optimal solution

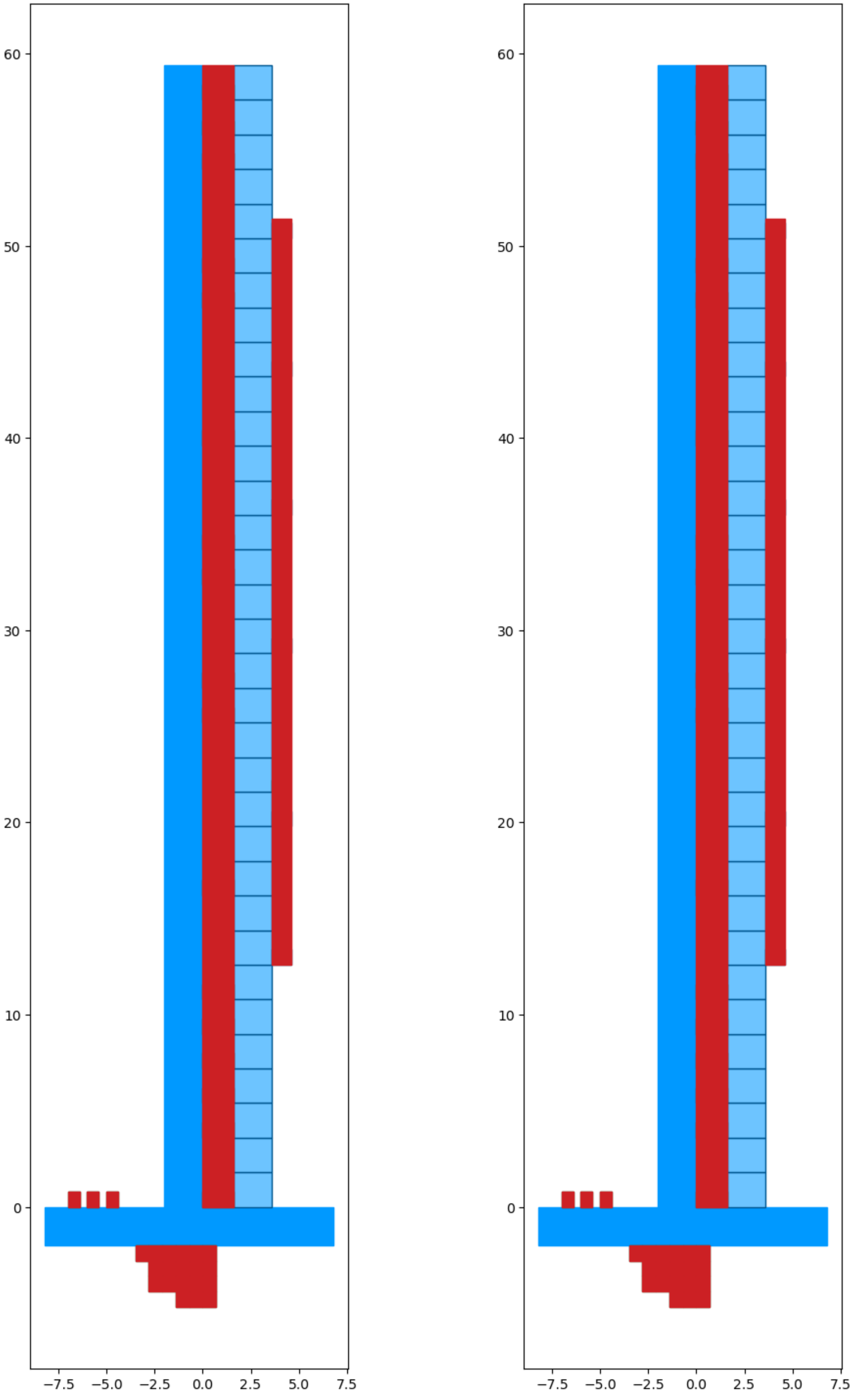
Figure B.15: System footprint of optimal solutions - experimental setup 7



(a) BOM optimal solution

(b) Floorspace optimal solution

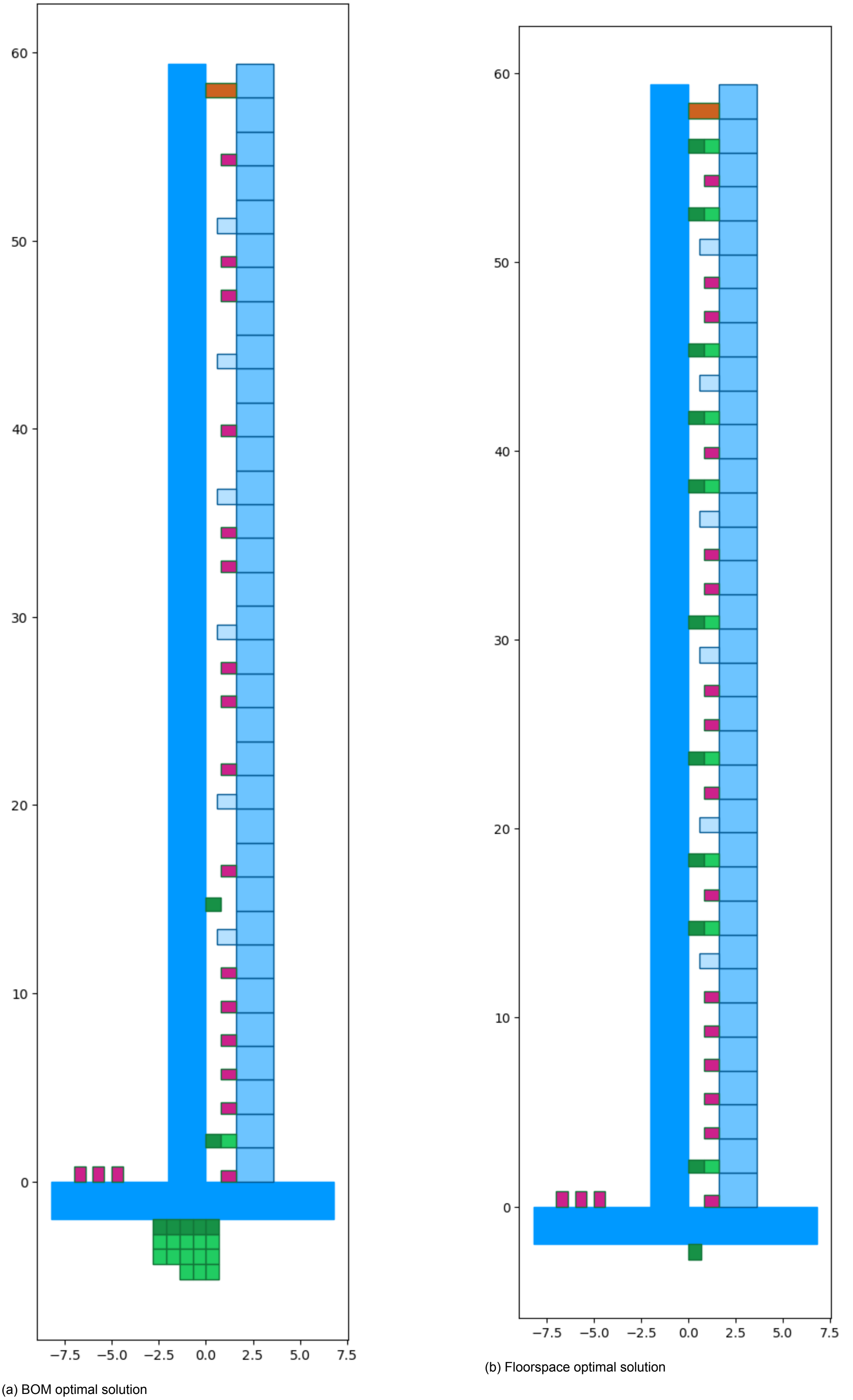
Figure B.16: Generated environment designs - experimental setup 8



(a) BOM optimal solution

(b) Floorspace optimal solution

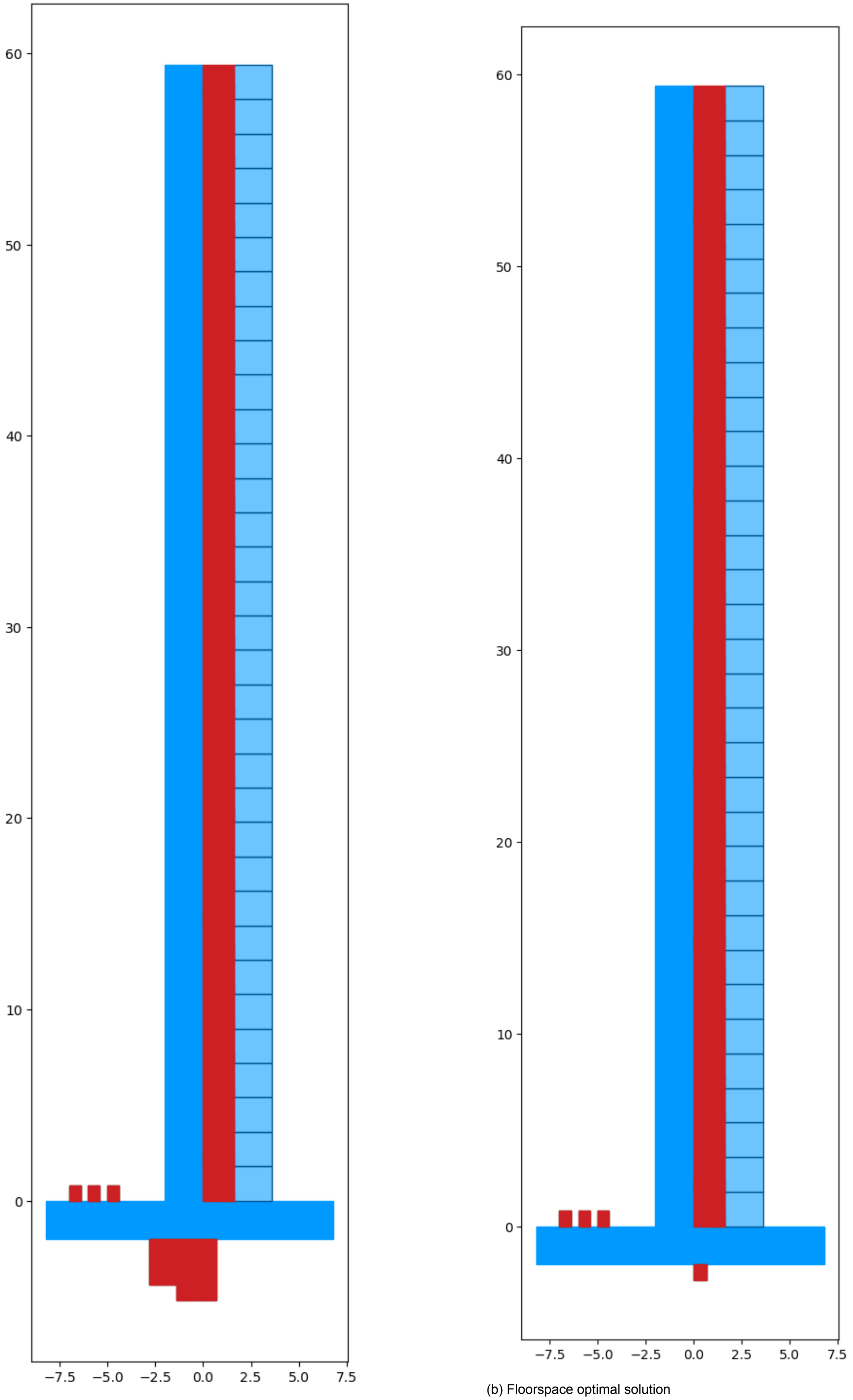
Figure B.17: System footprint of optimal solutions - experimental setup 8



(a) BOM optimal solution

(b) Floorspace optimal solution

Figure B.18: Generated environment designs - experimental setup 9



(a) BOM optimal solution

(b) Floorspace optimal solution

Figure B.19: System footprint of optimal solutions - experimental setup 9

C

Tables

Production cell	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Demand (in bins per shift)	5	11	41	13	13	82	82	20	31	25	49	4	4	6	16
Bin height (mm)	50	50	50	50	50	75	50	50	50	50	50	50	50	50	50
Demand (in stacks per shift)	1	2	6	2	2	17	12	3	5	4	7	1	1	1	3
Storage capabilities DSCAB	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Stack for MBC	0	1	5	1	1	16	11	2	4	3	6	0	0	0	2

Table C.1: Demand change due to storage location DSCABs usage - experimental setup storage capabilities DSCAB

Production cell	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Demand (in bins per shift)	5	11	41	13	13	82	82	20	31	25	49	4	4	6	16
Bin height (mm)	50	50	50	50	50	75	50	50	50	50	50	50	50	50	50
Demand (in stacks per shift)	1	2	6	2	2	17	12	3	5	4	7	1	1	1	3
Storage capabilities DSCAB	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Stack for MBC	0	0	3	0	0	14	9	0	2	1	4	0	0	0	0

Table C.2: Demand change due to storage location DSCABs usage - experimental setup enhanced storage capabilities DSCAB

C.1. Demand modification results - system footprint optimal solutions

	Total system footprint - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	98,43	98,43	98,43	98,43	98,43	133,7	99,55	148,51	98,97
60	98,99	98,43	98,43	98,99	98,99	134,82	100,11	149,07	99,53
50	101,79	99,55	98,43	101,79	101,79	139,04	104,03	151,87	102,33
43	103,47	101,23	98,43	103,47	104,03	140,72	104,59	153,55	104,01
35	106,83	105,15	101,79	106,83	107,39	144,08	107,95	156,91	107,37
33	109,5	108,38	103,9	109,5	110,06	145,76	110,62	159,58	109,05
30	110,62	108,94	106,7	111,18	111,74	146,88	111,74	160,7	110,17

	Increase in footprint in percentage - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	0%	0%	-1%	-1%	-1%	-1%	0%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	3%	1%	0%	3%	3%	3%	4%	2%	3%
43	5%	3%	0%	5%	5%	5%	4%	3%	5%
35	8%	7%	3%	8%	8%	8%	9%	5%	8%
33	11%	10%	6%	11%	11%	11%	11%	7%	10%
30	12%	11%	8%	12%	13%	13%	12%	8%	11%

	BOM cost increase w.r.t. 60s takt time - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-2%	-2%	0%	-2%	-7%	-3%	-1%	-2%	-3%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	6%	5%	5%	5%	18%	10%	27%	8%
43	9%	10%	8%	9%	10%	25%	11%	33%	14%
35	15%	17%	14%	15%	16%	35%	29%	45%	23%
33	22%	26%	19%	22%	23%	41%	38%	59%	28%
30	24%	27%	24%	25%	27%	46%	40%	64%	31%

	Total cost increase w.r.t. 60s takt time - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	0%	-1%	-5%	-2%	-1%	-1%	-2%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	5%	4%	5%	5%	12%	9%	14%	7%
43	8%	9%	6%	8%	9%	17%	10%	19%	11%
35	14%	16%	12%	14%	15%	24%	24%	26%	19%
33	20%	23%	17%	20%	21%	28%	31%	34%	23%
30	22%	24%	22%	23%	25%	32%	33%	37%	25%

	Floorspace increase w.r.t. case study - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	0%	0%	0%	0%	36%	6%	51%	1%
60	0%	-1%	-1%	0%	0%	36%	6%	51%	1%
50	0%	-2%	-3%	0%	0%	37%	7%	49%	1%
43	0%	-2%	-5%	0%	1%	36%	5%	48%	1%
35	0%	-2%	-5%	0%	1%	35%	6%	47%	1%
33	0%	-1%	-5%	0%	1%	33%	6%	46%	0%
30	0%	-2%	-4%	1%	1%	33%	6%	45%	0%

	BOM increase w.r.t. case study - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	-5%	12%	0%	-5%	-34%	-17%	-49%	-37%
60	0%	-5%	10%	0%	0%	-33%	-18%	-49%	-36%
50	0%	-4%	9%	0%	0%	-25%	-14%	-39%	-34%
43	0%	-4%	9%	0%	1%	-24%	-15%	-37%	-33%
35	0%	-3%	9%	0%	1%	-22%	-7%	-35%	-32%
33	0%	-2%	7%	0%	1%	-23%	-7%	-33%	-33%
30	0%	-2%	11%	1%	3%	-21%	-7%	-32%	-32%

	Total costs increase w.r.t. case study - Floorspace optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	-4%	7%	0%	-4%	-15%	-12%	-24%	-26%
60	0%	-4%	6%	0%	0%	-15%	-12%	-24%	-26%
50	0%	-4%	5%	0%	0%	-9%	-9%	-17%	-25%
43	0%	-4%	4%	0%	1%	-8%	-11%	-17%	-24%
35	0%	-3%	4%	0%	1%	-7%	-5%	-16%	-23%
33	0%	-2%	3%	0%	1%	-9%	-4%	-15%	-24%
30	0%	-2%	6%	1%	2%	-8%	-4%	-14%	-24%

C.2. Demand modification results - BOM optimal solutions

	Total system footprint - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	109,63	107,39	105,15	109,63	107,39	133,70	108,51	148,51	110,17
60	110,19	107,95	105,71	110,19	107,95	134,82	108,51	149,07	110,73
50	112,99	110,75	107,39	112,99	110,19	139,04	110,75	151,87	113,53
43	114,67	112,43	109,07	114,67	111,87	140,72	112,99	153,55	115,21
35	118,03	116,35	112,43	118,03	114,11	144,08	115,79	156,91	118,57
33	120,70	119,58	114,54	120,70	116,78	145,76	119,02	159,58	120,25
30	121,82	120,14	117,34	122,38	117,90	146,88	119,58	160,70	121,37

	Floorspace increase w.r.t. 60s takt time - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	-1%	-1%	-1%	-1%	0%	0%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	3%	3%	2%	3%	2%	3%	2%	2%	3%
43	4%	4%	3%	4%	4%	4%	4%	3%	4%
35	7%	8%	6%	7%	6%	7%	7%	5%	7%
33	10%	11%	8%	10%	8%	8%	10%	7%	9%
30	11%	11%	11%	11%	9%	9%	11%	8%	10%

	BOM increase w.r.t. 60s takt time - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	-2%	-2%	-1%	-3%	-1%	-2%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	6%	3%	5%	4%	18%	6%	27%	9%
43	9%	9%	6%	9%	7%	25%	11%	33%	14%
35	15%	17%	13%	15%	11%	35%	29%	45%	24%
33	23%	26%	18%	23%	18%	41%	38%	59%	30%
30	25%	27%	23%	26%	20%	46%	41%	64%	34%

	Total costs increase w.r.t. 60s takt time - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	-1%	-1%	-1%	-1%	-1%	-2%	-1%	-1%	-1%
60	0%	0%	0%	0%	0%	0%	0%	0%	0%
50	5%	5%	3%	5%	4%	12%	5%	14%	7%
43	8%	8%	6%	8%	6%	17%	9%	19%	11%
35	14%	15%	12%	14%	10%	24%	23%	26%	19%
33	20%	23%	16%	20%	16%	28%	30%	34%	23%
30	22%	24%	22%	23%	18%	32%	32%	37%	26%

	Floorspace increase w.r.t. case study - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	-2%	-4%	0%	-2%	22%	3%	35%	0%
60	0%	-2%	-4%	0%	-2%	22%	3%	35%	0%
50	0%	-2%	-5%	0%	-2%	23%	2%	34%	0%
43	0%	-2%	-5%	0%	-2%	23%	2%	34%	0%
35	0%	-1%	-5%	0%	-3%	22%	3%	33%	0%
33	0%	-1%	-5%	0%	-3%	21%	3%	32%	0%
30	0%	-1%	-4%	0%	-3%	21%	3%	32%	0%

	BOM increase w.r.t. case study - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	-4%	10%	-2%	-8%	-33%	-17%	-48%	-38%
60	0%	-4%	11%	-1%	-8%	-31%	-17%	-48%	-37%
50	0%	-4%	9%	-1%	-9%	-23%	-16%	-37%	-35%
43	0%	-4%	8%	-1%	-9%	-21%	-16%	-36%	-34%
35	0%	-3%	9%	-1%	-11%	-20%	-8%	-34%	-32%
33	0%	-2%	7%	-1%	-11%	-21%	-7%	-32%	-34%
30	0%	-3%	9%	0%	-11%	-20%	-7%	-31%	-33%

	Total costs increase w.r.t. case study - BOM optimal solution								
Takt Time	1	2	3	4	5	6	7	8	9
67	0%	-4%	5%	-1%	-6%	-16%	-12%	-25%	-26%
60	0%	-4%	5%	-1%	-6%	-15%	-12%	-24%	-26%
50	0%	-4%	3%	-1%	-7%	-9%	-12%	-18%	-25%
43	0%	-4%	3%	-1%	-8%	-8%	-11%	-17%	-24%
35	0%	-3%	4%	0%	-9%	-7%	-5%	-16%	-23%
33	0%	-1%	2%	0%	-9%	-9%	-5%	-15%	-24%
30	0%	-3%	5%	0%	-9%	-9%	-5%	-15%	-23%

D

Python code

D.1. Quantitative optimization model

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  8 15:06:06 2022

@author: TIMMAA
"""
#import gurobipy as GRB
from gurobipy import *
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd
import gurobipy as gp
from openpyxl import load_workbook


#%% Read Data from file
DataFile = r'C:\Users\TIMMAA\.spyder-py3\MBC_Optimization_InputFile.xlsx'

DataS = pd.read_excel(DataFile, sheet_name='Single␣data')
DataU = pd.read_excel(DataFile, sheet_name='Unit␣data')
DataP = pd.read_excel(DataFile, sheet_name='Product␣data')
DataE = pd.read_excel(DataFile, sheet_name='MBC␣Exception␣data')
DataA = pd.read_excel(DataFile, sheet_name='AGV␣data')

#%% Conditions
StoreFinishedProducts        = pd.DataFrame(DataS, columns=['SFP']).bool()
# True or False for condition
UseStorageInMBC              = pd.DataFrame(DataS, columns=['USIM']).bool()
# True or False for condition
FootprintTotal               = pd.DataFrame(DataS, columns=['WSF']).bool()
# True or False for condition
Optimization                 = pd.DataFrame(DataS, columns=['OPT']).OPT.item()
# True or False for condition


#%% Inputs
```

```
Bin50Size        = pd.DataFrame(DataS, columns=['50MM']).to_numpy()[:,0]
# The number of 50 mm bins in a stack
Bin75Size        = pd.DataFrame(DataS, columns=['75MM']).to_numpy()[:,0]
# The number of 75 mm bins in a stack
FTE_c            = pd.DataFrame(DataS, columns=['FTE']).to_numpy()[:,0]
# FTE Costs per day
FloorspaceCosts = pd.DataFrame(DataS, columns=['FLOORSPACE']).to_numpy()[:,0]
# Costs of m2 floorspace per day
OD               = pd.DataFrame(DataS, columns=['OD']).to_numpy()[:,0]
# Operating Days (in days)
MaxMBClength     = pd.DataFrame(DataS, columns=['MaxMBC']).to_numpy()[:,0]
# Maximum number of MBC slaves behind the master
BufferTime       = pd.DataFrame(DataS, columns=['BufTime']).to_numpy()[:,0]
# How long does the system have to run withour restocking? (hours)
TaktTime         = pd.DataFrame(DataS, columns=['TaktTime']).to_numpy()[:,0]
# Takt time for one product
FPPB             = pd.DataFrame(DataS, columns=['FPPB']).to_numpy()[:,0]
# Finished products per bin

UnitNames        = pd.DataFrame(DataU, columns=['Name']).values.tolist()
# List of headers of unit data
Units_init       = pd.DataFrame(DataU, columns=['Units_init']).to_numpy()
# Initial amount of units
BOM              = pd.DataFrame(DataU, columns=['BOM']).to_numpy()
# Bill of material for each unit
AssemblyTime     = pd.DataFrame(DataU, columns=['ASST']).to_numpy()
# Assembly time in days
InstallationTime = pd.DataFrame(DataU, columns=['INST']).to_numpy()
# Installation time in days
Footprint        = pd.DataFrame(DataU, columns=['FOOTPRINT']).to_numpy()
# Footprint of each machine in m2
SI               = pd.DataFrame(DataU, columns=['SI']).to_numpy()[:,0]
# Stack locations per unit

ProdCell         = pd.DataFrame(DataP, columns=['ProdCell']).to_numpy()[:,0]
# Height of the bins for product i
ProdID_import    = pd.DataFrame(DataP, columns=['ProdID']).to_numpy()[:,0]
# Height of the bins for product i
BinHeight        = pd.DataFrame(DataP, columns=['BinHeight']).to_numpy()[:,0]
# Height of the bins for product i
Demand_import    = pd.DataFrame(DataP, columns=['Demand']).to_numpy()[:,0]
# Demand in bins for product i in I for one shift

MBCException     = pd.DataFrame(DataE, columns=['Exception']).to_numpy()[:,0]
# Data of the exceptions on the MBC grid max values

AGVTravelInfo_headers    = pd.DataFrame(DataA, columns=['Headers']).values.tolist()
# Headers for the AGV info
ATI                      = pd.DataFrame(DataA, columns=['Times']).to_numpy()[:,0]
# AGV Travel Info (see headers above)
IntroMBC                 = pd.DataFrame(DataA, columns=['IntroMBC']).to_numpy()[:,0]
# Number of actions described by AGV Travel Info Headers for job: From intro to MBC
MBCDSCAB                 = pd.DataFrame(DataA, columns=['MBCDSCAB']).to_numpy()[:,0]
# Number of actions described by AGV Travel Info Headers for job: From MBC to DSCAB
FinishMBC                = pd.DataFrame(DataA, columns=['FinishMBC']).to_numpy()[:,0]
# Number of actions described by AGV Travel Info Headers for job: From Finished produ
```

```
#%% Rewrite inputs into right matrices and data types. Delete zeros from inputs.

ProdID_import = [i for i in ProdID_import if i != 0]
# Delete zero inputs (non used cells)
Demand_import = [i for i in Demand_import if i != 0]
# Delete zero inputs (non used cells)

Demand = np.zeros_like(ProdID_import)
# Initialize demand vector
ProdID = np.arange(len(ProdID_import))+1
# Create product ID vector

for j in range(len(Demand)):
# Gather demand for product i in cell j for all products i. Usage: if multiple cell need t
    for i in range(len(Demand)):
        if ProdID[j] == ProdID_import[i]:
# Find product IDs not on ID's index in vector
            Demand[j] = Demand[j] + Demand_import[i]
# Add the demand of those IDs to the demand vector

Bas = np.zeros(len(ProdID))
# Amount of bins in a stack of product i
for i in range(len(ProdID)):
# Couple binsize to number of bins of that size in a stack
    if BinHeight[i] == 50:
        Bas[i] = Bin50Size[0]
    if BinHeight[i] == 75:
        Bas[i] = Bin75Size[0]

if Optimization == 'Footprint':
# If optimizing for footprint, wasted space must be considered.
    FootprintTotal = True

#%% Store finished products criterea check

BufferModifier = BufferTime / 8
# BufferTime base value is 8 hours. Input divided by 8 gives the fraction to modify by.

D = Demand * BufferModifier
# Buffer modifier applied to the demand vector

if StoreFinishedProducts == True:
# Number of stored finished product is calculatedfrom the takttime of a product
    BufTime = 3600 * BufferTime
# Buffertime in seconds
    FinishedBins = (BufTime/TaktTime)/(FPPB)
# Produced product in buffertime divided by the numbr of finished product per bin
    ProdID = np.append(ProdID,ProdID[-1]+1)
# Add finished product to product ID vector
    D = np.append(D,FinishedBins)
# Add finished product demand to storage demand vector D
    Bas = np.append(Bas,5)
# Add finished product bins per stack to bins per stack vector Bas
```

```
#%% Optional demand modifier, base value 1
Change = 1.0
D = D * Change

#%% ----- Code support parameters -----
PrevCost = 99999999999
ObjValueHeaders = ['Total','BOM','Assembly','Installation','Floorspace']
ObjValue = np.zeros([6,len(D)])

#%%  ------ Sets ------
I = range (len(D))              # Set of different products
J = range (len(Units_init))            # Set of different storage units
L = range (Sl[1])               # Set of number of storage locations of a MBC (vertic
N = range (len(ATI))  # Set of different info points of an AGV

#%%  Bin supply needed per product ID
Bs = np.zeros_like(D)
# Number of stacks of bins of product i in I
if UseStorageInMBC == True:
    for i in range(np.size(D)):
        Bs[i] = math.ceil(D[i]/Bas[i]) - math.ceil(Sl[4]/2)
# Number of stacks of bins of product i in I minus one location for empty bin storage
else:
    for i in range(np.size(D)):
        Bs[i] = math.ceil(D[i]/Bas[i])
# Number of stacks of bins of product i in I

#%% AGV travel times          Data gathered from : RSD6996203261R04.pdf
AvailableTime = BufferTime * 3600
# The amount of time a single AGV is available. Equal to the buffer time, charging ti

if StoreFinishedProducts == True:
# If finished product need to be stored, an extra job is added
    AGVtimes = np.zeros([3,len(N)])
    for n in N:
        AGVtimes[0,n] = IntroMBC[n]*ATI[n]*(sum(Bs) - Bs[-1])
        AGVtimes[1,n] = MBCDSCAB[n]*ATI[n]*(sum(Bs) - Bs[-1])
        AGVtimes[2,n] = FinishMBC[n]*ATI[n]*Bs[-1]
else:
    AGVtimes = np.zeros([2,len(N)])
    for n in N:
        AGVtimes[0,n] = IntroMBC[n]*ATI[n]*sum(Bs)
        AGVtimes[1,n] = MBCDSCAB[n]*ATI[n]*sum(Bs)

ActiveTime = sum(sum(AGVtimes))
# Total time an AGV is needed to be active in the environment.

AGVreq = math.ceil(ActiveTime/AvailableTime)
# Total number AGVs needed in the environment

#%% MBC row exception

MBCException = [i for i in MBCException if i != 0]
if FootprintTotal == True:
    MBCmax = np.full((1,(len(D))),MaxMBClength)
    for e in range(len(MBCException)):
        MBCmax[0][e] = MBCException[e]
#else:
#    MBCmax = np.full((1,(len(D))),MaxMBClength)

K = range (max(MBCmax[0]))          # Set of maximum number of MBCs to be placed behind each

#%% Model part

for Mmax in range(len(MBCException)+1,9 ):#len(D)):
    model = Model ('currentstate')
    print('Current□width□of□m□is:' , Mmax)
    M = range (Mmax)                # Large enough set of different places for MBCs for the


    #%%  ------ Decision variables ------

    # Variable Xunits (Array with integers indicating how much of each unit are used)
    Xunits = {}
    for j in J:
        Xunits[j] = model.addVar (lb = 0, vtype = GRB.INTEGER, name= 'Xunits[' + str(j) +

    # The grid of the MBCs and their stack locations
    # X-axis = vertical amount of storage locations in an MBC (4 in the current unit)
    # Y-axis = horizontal number of MBC units that are next to each other
    # Z-axis = depth of the MBC combiination (amount of slaves behind each master)
    MBCGrid = {}
    MBCusl = {}
    Assign = {}
    for l in L:
        for m in M:
            MBCusl[l,m] = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'MBCuls[' +
# Unique storage location sizes (in stacks) and their places projected on the MBC grid
            for k in K:
                MBCGrid[l,m,k] = model.addVar (lb = 0, vtype = GRB.BINARY, name = 'MBCGric
            for i in I:
                Assign[l,m,i] = model.addVar (lb = 0,vtype = GRB.BINARY, name = 'Assign['



    #%% Costs variables

    # Costs per unit (BOM)
    Cb = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'Cb[]')

    # Costs for assamble
    Ca = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'Ca[]')

    # Costs for installation
    Ci = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'Ci[]')

    # Costs for floorspace
    Cf = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'Cf[]')

    # Total variable costs (time depend costs) per unit type j
    Ct = model.addVar (lb = 0, vtype = GRB.CONTINUOUS, name = 'Ct[]')
```

```
# Maximum depth of Grid
GridMax = model.addVar(lb = 0, vtype = GRB.CONTINUOUS, name = 'GridMax')

# Integrate new variables
model.update ()


#%%  ------ Cost functions and objective function ------

# Ct (total costs) is calculated by adding total fixed and variable costs
#Ct = quicksum(Cb[j] for j in J) + quicksum(Ct_v[j] for j in J)
if Optimization == 'Footprint':
    model.setObjective(Cf)
if Optimization == 'BOM':
    model.setObjective(Cb)
if Optimization == 'Total':
    model.setObjective(Ct)
model.modelSense = GRB.MINIMIZE
model.update()


#%%  --- General constraints ---

con1 = {}       # Constraint 1: Fill MBCGrid in the l direction (heigth) since ar
con2 = {}       # Constraint 2: There is always an unit before another unit (no g
con3 = {}       # Constraint 3: The numer of stack locations in an USL is the num
con10 = {}      # Constraint 10: One USL can only hold one product ID
for l in L:
    for m in M:
        con3[l,m] = model.addConstr((MBCusl[l,m] == quicksum(MBCGrid[l,m,k] for k
        con10[l,m] = model.addConstr((quicksum(Assign[l,m,i] for i in I) <=1))
        for k in K:
            con1[m] = model.addConstr(MBCGrid[l,m,k] == MBCGrid[0,m,k])
        for k in range(0,len(K)-1):
            con2[l,m,k] = model.addConstr((MBCGrid[l,m,k] >= MBCGrid[l,m,(k+1)]))

con11 = {}      # Constraint 11: Quantity of all storage locations of product i s
for i in I:
    con11[i] = model.addConstr((quicksum(Assign[l,m,i] * MBCusl[l,m] for l in L f

con4 = {}       # Constraint 4: Constraint enabling MBC exceptions
for m in M:
    con4[m] = model.addConstr(quicksum(MBCGrid[0,m,k] for k in K) <= MBCmax[0][m]

#%%  --- Unit count constraints ---

# Constraint 30: Set up MBCGrid front row, amount of MBC masters un MBCGrid equa
con30 = model.addConstr((Xunits[1]) == quicksum(MBCGrid[0,m,0] for m in M))

# Constraint 31: Count the amount of MBCs and subtract number of Masters (MBC HB
con31 = model.addConstr(Xunits[2] == (quicksum(MBCGrid[0,m,k] for m in M for k in

# Constraint 33: Number of required (SPEL) introduction stations
con33 = model.addConstr((Xunits[0] == Units_init[0]) ,'con33[]')

# Constraint 34: Number of required MBC back docks  (MBC Slaves)
```

```
    con34 = model.addConstr((Xunits[3] == Units_init[3]) ,'con34[]')

    # Constraint 35: Number of required DSCABs
    con35 = model.addConstr((Xunits[4] == Units_init[4]) ,'con35[]')

    # Constraint 36: Number of required AGVs
    con36 = model.addConstr((Xunits[5] == AGVreq))
#!!!!!

    #%%  --- Cost constraints ---
    # Constraint 40: BOM costs of units equals Xunits[j] times BOM[j]
    con40 = model.addConstr(Cb == quicksum((Xunits[j] * BOM[j,0])for j in J) )

    # Constraint 41: Assembly costs
    con41 = model.addConstr(Ca == quicksum(Xunits[j] * FTE_c * AssemblyTime[j] for j in J)

    # Constraint 42: Installation costs
    con42 = model.addConstr(Ci == quicksum(Xunits[j]*InstallationTime[j]*FTE_c for j in J)

    # Constraint 44: Max value of MBC usl
    con44 = model.addConstr(GridMax == max_(MBCusl[0,m] for m in M))

    # Support for constraint 45. Set P to slice off grid where exception begin (included i
    P = range(0,len(MBCException))

    if FootprintTotal == True and len(MBCException) == 0:
        # Constraint 46: calculate floorspace costs
        con46 = model.addConstr(Cf == FloorspaceCosts[0] * OD[0]* (96 + quicksum(MBCGrid[0
    elif FootprintTotal == True and len(MBCException) != 0:
        con46 = model.addConstr(Cf == FloorspaceCosts[0] * OD[0]* (96 + quicksum(MBCGrid[0
    else:
        con46 = model.addConstr(Cf == FloorspaceCosts[0] * OD[0]* quicksum(Xunits[j] * Fo

    # Constraint 48: Total cost
    con48 = model.addConstr(Ct == Cb + Ca + Ci + Cf)

    #%%  ------ Optimize ------
    model.update()

    model.setParam( 'OutputFlag', False) # silencing gurobi output or not
    model.setParam ('MIPGap', 0);        # find the optimal solution
    model.write("output.lp")             # print the model in .lp format file
    model.setParam('TimeLimit',500)
    model.optimize ()

    #%%    ------- Break statements for the loop

    if model.status == GRB.Status.OPTIMAL:
        print('For␣M␣is', Mmax, '␣the␣solution␣is␣is:',model.objval)
        SQM = Cf.x / (FloorspaceCosts[0] * OD[0])
        ObjValue[0,Mmax] = Ct.x
        ObjValue[1,Mmax] = Cb.x
        ObjValue[2,Mmax] = Ca.x
        ObjValue[3,Mmax] = Ci.x
        ObjValue[4,Mmax] = Cf.x
        ObjValue[5,Mmax] = SQM
        if PrevCost <= model.objval+10:
            print('We␣zijn␣klaar␣en␣Prevcost␣is',PrevCost,'␣en␣objval␣is',model.objva
            break
        PrevCost = model.objval

    #%%  ------ Print calcualtions ------

if model.status == GRB.Status.OPTIMAL:
    Grid = np.zeros((len(L),len(M),len(K)))
    for l in L:
        for m in M:
            for k in K:
                Grid[l,m,k] = round(MBCGrid[l,m,k].x)


    GridUsl = np.zeros((len(L),len(M)))
    for l in L:
        for m in M:
                GridUsl[l,m] = round(MBCusl[l,m].x)

    Assign2 = np.zeros((len(L),len(M),len(I)))
    for l in L:
        for m in M:
            for i in I:
                Assign2[l,m,i] = round(Assign[l,m,i].x)

    Units = np.zeros(len(J))
    for j in J:
        Units[j] = round(Xunits[j].x)

Assign3 = np.zeros((len(L),len(M)))
for l in L:
    for m in M:
        Assign3[l,m] = Assign2[l,m,:].tolist().index(1) +1
        if GridUsl[l,m] == 0:
            Assign3[l,m] = 0



#   ------ Print calcualtions ------

if model.status == GRB.Status.OPTIMAL:
    print("--------------␣Results␣----------------")
    print('Best␣solution␣is:␣', model.objval , 'euro')
    print('BOM␣Costs␣are:␣', Cb.x , 'euro')
    #print(model.objval)
    print('--------------------------------')
    print('Unique␣storage␣locations␣projected␣on␣the␣front␣view␣of␣the␣MBC␣grid')
    print(GridUsl)
    print('--------------------------------')
    print('Number␣of␣AGV␣decided␣by␣model␣equals:', Xunits[5].x)
    print('--------------------------------')
    print('The␣units␣needed␣are:')
    print(UnitNames)
    print(Units)
```

```
fn = r'C:\Users\TIMMAA\.spyder-py3\Geometry.xlsx'

book = load_workbook(fn)

writer = pd.ExcelWriter(fn, engine='openpyxl')

writer.book = book
writer.sheets = dict((ws.title, ws) for ws in book.worksheets)

Rgrid.to_excel(writer, sheet_name='Results␣grid', header=None, index=False,
            startcol=1, startrow=2)

Rzeros.to_excel(writer, sheet_name='Results␣grid', header=None, index=False,
            startcol=1, startrow=10)

Rassign.to_excel(writer, sheet_name='Results␣grid', header=None, index=False,
            startcol=1, startrow=10)

writer.save()
writer.close()
#%% Return results to excel
```

D.2. Design generation code

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  8 15:06:06 2022

@author: TIMMAA
"""
#import gurobipy as GRB
from gurobipy import *
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd
import gurobipy as gp
from openpyxl import load_workbook

#%% Conditions

for i in range(20):
    print('Floorplan␣for␣costs␣of␣floorspace␣optimization␣[C/F]?')
    InputCheck = 'C'
    if InputCheck == 'C' or 'F' or 'c' or 'f':
        print('Lets␣go')
        break
    print('Please␣state␣only␣a␣[C]␣of␣a␣[F]')


Case                    = True
DSCAB_S                 = False
DSCAB_SE                = False
Shared                  = False
Shared_NL               = False
DSCAB_C                 = False
```

```
DSCAB_MOD                = False
DSCAB_MOD_NO             = False
AGV_Stack                = False

ShowIndividual           = False


if InputCheck == ’C’ or ’c’:
    CostOpt              = False
else:
    CostOpt              = False

#%% Data import
if Case == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Cas
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Cas

if DSCAB_S == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC

if DSCAB_SE == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC

if Shared == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Sha
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Sha

if Shared_NL == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Sha
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’Sha

if DSCAB_C == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC

if DSCAB_MOD == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder−py3\Geo.xlsx’, sheet_name=’DSC

if DSCAB_MOD_NO == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder-py3\Geo.xlsx’, sheet_name=’DSCAB-MC
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder-py3\Geo.xlsx’, sheet_name=’DSCAB-MC

if AGV_Stack == True:
    if CostOpt == True:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder-py3\Geo.xlsx’, sheet_name=’AGV-S-C’
    else:
        Data = pd.read_excel(r’C:\Users\TIMMAA\.spyder-py3\Geo.xlsx’, sheet_name=’AGV-S-F’


DataC = pd.read_excel(r’C:\Users\TIMMAA\.spyder-py3\Geo.xlsx’, sheet_name=’Collor’)


#%% Collor import
C_walkways = pd.DataFrame(DataC, columns=[’Walkways’]).Walkways.item()

C_walkways_E = pd.DataFrame(DataC, columns=[’WWE’]).WWE.item()

C_Cells = pd.DataFrame(DataC, columns=[’Cells’]).Cells.item()

C_Liquid = pd.DataFrame(DataC, columns=[’Liquid’]).Liquid.item()

C_Borders_L = pd.DataFrame(DataC, columns=[’BordersL’]).BordersL.item()

C_SPEL = pd.DataFrame(DataC, columns=[’SPEL’]).SPEL.item()

C_Master = pd.DataFrame(DataC, columns=[’Master’]).Master.item()

C_HBT = pd.DataFrame(DataC, columns=[’HBT’]).HBT.item()

C_Slave = pd.DataFrame(DataC, columns=[’Slave’]).Slave.item()

C_DSCAB = pd.DataFrame(DataC, columns=[’DSCAB’]).DSCAB.item()

C_DSCAB_Move_M = pd.DataFrame(DataC, columns=[’DSCABmoveM’]).DSCABmoveM.item()

C_DSCAB_Move = pd.DataFrame(DataC, columns=[’DSCABmove’]).DSCABmove.item()

C_Borders_U = pd.DataFrame(DataC, columns=[’BordersU’]).BordersU.item()

C_Indicate = pd.DataFrame(DataC, columns=[’Indicate’]).Indicate.item()

C_Arrow = pd.DataFrame(DataC, columns=[’Arrow’]).Arrow.item()

C_Borders_Mob = pd.DataFrame(DataC, columns=[’BordersMob’]).BordersMob.item()

#%% Import locations
Cells = pd.DataFrame(Data, columns=[’Ypos’]).to_numpy()[:,0]
Cells = Cells[~np.isnan(Cells)]

Lengte = pd.DataFrame(Data, columns=[’Lengte’]).to_numpy()[:,0]
Lengte = Lengte[~np.isnan(Lengte)]

Liquid_y = pd.DataFrame(Data, columns=[’Liquid’]).to_numpy()[:,0]
Liquid_y = Liquid_y[~np.isnan(Liquid_y)]

# DSCABs
DSCAB_y  = pd.DataFrame(Data, columns=['DSCAB']).to_numpy()[:,0]
DSCAB_y = DSCAB_y[~np.isnan(DSCAB_y)]

DSCABVx  = pd.DataFrame(Data, columns=['DSCABVx']).to_numpy()[:,0]
DSCABVx = DSCABVx[~np.isnan(DSCABVx)]

DSCABVy  = pd.DataFrame(Data, columns=['DSCABVy']).to_numpy()[:,0]
DSCABVy = DSCABVy[~np.isnan(DSCABVy)]

# DSCABS mobile
DSCAB_MM_y  = pd.DataFrame(Data, columns=['DSCABMM']).to_numpy()[:,0]
# Master locations of Mobile DSCAB
DSCAB_MM_y = DSCAB_MM_y[~np.isnan(DSCAB_MM_y)]

DSCAB_MS_y  = pd.DataFrame(Data, columns=['DSCABMS']).to_numpy()[:,0]
# Slave locaion of Mobile DSCAB
DSCAB_MS_y = DSCAB_MS_y[~np.isnan(DSCAB_MS_y)]

# SPELs
SPELH = pd.DataFrame(Data, columns=['SPELH']).to_numpy()[:,0]
SPELH = SPELH[~np.isnan(SPELH)]

SPELVx = pd.DataFrame(Data, columns=['SPELVx']).to_numpy()[:,0]
SPELVx = SPELVx[~np.isnan(SPELVx)]

SPELVy = pd.DataFrame(Data, columns=['SPELVy']).to_numpy()[:,0]
SPELVy = SPELVy[~np.isnan(SPELVy)]

# MBCs
MBCMVx = pd.DataFrame(Data, columns=['MBCMVx']).to_numpy()[:,0]
MBCMVx = MBCMVx[~np.isnan(MBCMVx)]

MBCMVy = pd.DataFrame(Data, columns=['MBCMVy']).to_numpy()[:,0]
MBCMVy = MBCMVy[~np.isnan(MBCMVy)]

MBChbtVx = pd.DataFrame(Data, columns=['MBChbtVx']).to_numpy()[:,0]
MBChbtVx = MBChbtVx[~np.isnan(MBChbtVx)]

MBChbtVy = pd.DataFrame(Data, columns=['MBChbtVy']).to_numpy()[:,0]
MBChbtVy = MBChbtVy[~np.isnan(MBChbtVy)]

MBCMHx = pd.DataFrame(Data, columns=['MBCMHx']).to_numpy()[:,0]
MBCMHx = MBCMHx[~np.isnan(MBCMHx)]

MBCMHy = pd.DataFrame(Data, columns=['MBCMHy']).to_numpy()[:,0]
MBCMHy = MBCMHy[~np.isnan(MBCMHy)]

MBChbtHx = pd.DataFrame(Data, columns=['MBChbtHx']).to_numpy()[:,0]
MBChbtHx = MBChbtHx[~np.isnan(MBChbtHx)]

MBChbtHy = pd.DataFrame(Data, columns=['MBChbtHy']).to_numpy()[:,0]
MBChbtHy = MBChbtHy[~np.isnan(MBChbtHy)]
```

```
MBCSVx = pd.DataFrame(Data, columns=['MBCSVx']).to_numpy()[:,0]
MBCSVx = MBCSVx[~np.isnan(MBCSVx)]

MBCSVy = pd.DataFrame(Data, columns=['MBCSVy']).to_numpy()[:,0]
MBCSVy = MBCSVy[~np.isnan(MBCSVy)]


#%% Plotting
fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)
fig.set_size_inches(10.5, 18.5, forward=True)
fig.set_dpi(100)

#%% Production Line

# Walkways
WalkwayLongV = plt.Rectangle((-2,0), 2, (Lengte+1.8), fc= C_walkways,ec=C_walkways)
plt.gca().add_patch(WalkwayLongV)

WalkwayLongH = plt.Rectangle((-8.2,-2), 15, 2, fc= C_walkways,ec=C_walkways)
plt.gca().add_patch(WalkwayLongH)

if DSCAB_C == True:
    WalkwayLongVE = plt.Rectangle((-2.3,0), 0.3, (Lengte+1.8), fc= C_walkways_E,ec=C_walkw
    plt.gca().add_patch(WalkwayLongVE)

    WalkwayShortV = plt.Rectangle((-8.2,-7.2), 2, 5.2, fc= C_walkways,ec=C_walkways)
    plt.gca().add_patch(WalkwayShortV)

    WalkwayShortH = plt.Rectangle((-6.2,-7.2), 7, 2, fc= C_walkways,ec=C_walkways)
    plt.gca().add_patch(WalkwayShortH)

# Cells
for i in range(len(Cells)):
    Cells_p = plt.Rectangle((1.6,Cells[i]), 2, 1.8, fc=C_Cells,ec=C_Borders_L)
    plt.gca().add_patch(Cells_p)

# Liquids
if DSCAB_MOD_NO == True:
    for i in range(len(Liquid_y)):
        Liquid_p = plt.Rectangle((3.6,Liquid_y[i]), 1, 0.8, fc=C_Liquid,ec=C_Borders_L)
        plt.gca().add_patch(Liquid_p)
else:
    for i in range(len(Liquid_y)):
        Liquid_p = plt.Rectangle((0.6,Liquid_y[i]), 1, 0.8, fc=C_Liquid,ec=C_Borders_L)
        plt.gca().add_patch(Liquid_p)

#%% Units



# DSCABs horizontal
for i in range(len(DSCAB_y)):
    DSCAB_p = plt.Rectangle((0,DSCAB_y[i]), 1.6, 0.8, fc= C_DSCAB,ec=C_Borders_U)
    plt.gca().add_patch(DSCAB_p)
```

```
# DSCABs horizontal Mobile Master Location
for i in range(len(DSCAB_MM_y)):
    DSCAB_MM_p = plt.Rectangle((0,DSCAB_MM_y[i]), 1.6, 0.8, fc= C_DSCAB_Move_M,ec=C_
    plt.gca().add_patch(DSCAB_MM_p)

# DSCABs horizontal Mobile Slave locations
for i in range(len(DSCAB_MS_y)):
    DSCAB_MS_p = plt.Rectangle((0,DSCAB_MS_y[i]), 1.6, 0.8, fc= C_DSCAB_Move,ec=C_Bor
    plt.gca().add_patch(DSCAB_MS_p)

# DSCABs verical
for i in range(len(DSCABVy)):
    DSCABV_p = plt.Rectangle((DSCABVx[i],DSCABVy[i]), 0.8, 1.6, fc= C_DSCAB,ec=C_Bor
    plt.gca().add_patch(DSCABV_p)

# Spel Horizontaal
if DSCAB_MOD_NO == True:
    for i in range(len(SPELH)):
        SPELH_p = plt.Rectangle((3.6,SPELH[i]), 0.8, 0.6, fc= C_SPEL ,ec=C_Borders_U)
        plt.gca().add_patch(SPELH_p)
else:
    for i in range(len(SPELH)):
        SPELH_p = plt.Rectangle((0.8,SPELH[i]), 0.8, 0.6, fc= C_SPEL ,ec=C_Borders_U)
        plt.gca().add_patch(SPELH_p)

# Spel Vertical
for i in range(len(SPELVx)):
    SPELV_p = plt.Rectangle((SPELVx[i],SPELVy[i]), 0.6, 0.8, fc= C_SPEL ,ec=C_Borders
    plt.gca().add_patch(SPELV_p)

# MBC Master V
for i in range(len(MBCMVx)):
    MBCMV_p = plt.Rectangle((MBCMVx[i],MBCMVy[i]), 0.7, 0.8, fc=C_Master,ec=C_Borders
    plt.gca().add_patch(MBCMV_p)

# MBC HBT V
for i in range(len(MBChbtVx)):
    MBChbtV_p = plt.Rectangle((MBChbtVx[i],MBChbtVy[i]), 0.7, 0.8, fc=C_HBT,ec=C_Bor
    plt.gca().add_patch(MBChbtV_p)

# MBC Slave V
for i in range(len(MBCSVx)):
    MBCSV_p = plt.Rectangle((MBCSVx[i],MBCSVy[i]), 0.7, 0.8, fc=C_Slave,ec=C_Borders_
    plt.gca().add_patch(MBCSV_p)

# MBC Master H
for i in range(len(MBCMHx)):
    MBCMH_p = plt.Rectangle((MBCMHx[i],MBCMHy[i]), 0.8, 0.7, fc=C_Master,ec=C_Borders
    plt.gca().add_patch(MBCMH_p)

# MBC HBT H
for i in range(len(MBChbtHx)):
    MBChbtH_p = plt.Rectangle((MBChbtHx[i],MBChbtHy[i]), 0.8, 0.7, fc=C_HBT,ec=C_Bor
    plt.gca().add_patch(MBChbtH_p)
```

```
# Arrows
if DSCAB_MOD == True:
    plt.arrow(0.8, 0.4, 0, 10.4,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arr
# Cluster 1
    plt.arrow(0.8, 25.6, 0, 1.4,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arr
# Cluster 2
    plt.arrow(0.8, 32.8, 0, 1.4,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arr
# Cluster 3
    plt.arrow(0.8, 47.2, 0, 1.4,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arr
# Cluster 4

if DSCAB_MOD_NO == True:
    plt.arrow(0.8, 0.4, 0, 25,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arrow
# Cluster 1
    plt.arrow(0.8, 47.2, 0, -19.6,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_
# Cluster 2
    plt.arrow(0.8, 47.2, 0, 6.8,head_width = 0.5, head_length = 0.5, fc= C_Arrow ,ec=C_Arr
# Cluster 2


#%% Wasted space
if ShowIndividual == False:
    # DSCAB area
    Indicate_p = plt.Rectangle((0,0), 1.6, (Lengte+1.8), fc= C_Indicate ,ec=C_Indicate)
    plt.gca().add_patch(Indicate_p)

        # MBC Master V
    for i in range(len(MBCMVx)):
        MBCMV_p = plt.Rectangle((MBCMVx[i],MBCMVy[i]), 0.7, 0.8, fc=C_Indicate,ec=C_Indica
        plt.gca().add_patch(MBCMV_p)

    # MBC HBT V
    for i in range(len(MBChbtVx)):
        MBChbtV_p = plt.Rectangle((MBChbtVx[i],MBChbtVy[i]), 0.7, 0.8, fc=C_Indicate,ec=C_
        plt.gca().add_patch(MBChbtV_p)

    # MBC Slave V
    for i in range(len(MBCSVx)):
        MBCSV_p = plt.Rectangle((MBCSVx[i],MBCSVy[i]), 0.7, 0.8, fc=C_Indicate,ec=C_Indica
        plt.gca().add_patch(MBCSV_p)

    if DSCAB_C == True:
        WalkwayLongVE = plt.Rectangle((-2.3,0), 0.3, (Lengte+1.8), fc=C_Indicate,ec=C_Ind
        plt.gca().add_patch(WalkwayLongVE)

        WalkwayShortV = plt.Rectangle((-8.2,-7.2), 2, 5.2, fc=C_Indicate,ec=C_Indicate)
        plt.gca().add_patch(WalkwayShortV)

        WalkwayShortH = plt.Rectangle((-6.2,-7.2), 7, 2, fc=C_Indicate,ec=C_Indicate)
        plt.gca().add_patch(WalkwayShortH)

        WastedSpace = plt.Rectangle((-6.2,-5.2),7, 3.2, fc=C_Indicate,ec=C_Indicate)
        plt.gca().add_patch(WastedSpace)

    if DSCAB_MOD_NO == True:
        Indicate_obstacles = plt.Rectangle((3.6,Liquid_y[0]), 1, (Liquid_y[-1]-Liquid_y[0]
), fc= C_Indicate ,ec=C_Indicate)
        plt.gca().add_patch(Indicate_obstacles)

    # Spel Vertical
    for i in range(len(SPELVx)):
        SPELV_p = plt.Rectangle((SPELVx[i],SPELVy[i]), 0.6, 0.8, fc= C_Indicate ,ec=C
        plt.gca().add_patch(SPELV_p)




#%% Show
plt.axis('scaled')
fig.savefig('test2png.png', dpi=100)

plt.show()
```

Bibliography

Aarts, A. (2010). Methodology of mathematical modeling.

Adby, P. (2013). *Introduction to optimization methods*. Springer Science & Business Media.

Babuska, I., & Oden, J. T. (2004). Verification and validation in computational engineering and science: Basic concepts. *Computer methods in applied mechanics and engineering*, *193*(36-38), 4057–4066.

Bahrami, B., Piri, H., & Aghezzaf, E.-H. (2019). Class-based storage location assignment: An overview of the literature. *16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, 390–397.

Baral, A. (2021). The 'design to cost' perspective.

Barasov, A. (1961). *Linear programming*. Fizmatgiz.

Chu, P. C., & Beasley, J. E. (1997). A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, *24*(1), 17–23.

De Koster, M., & Neuteboom, A. (2001). The logistics of supermarket chains. *Doetinchem, The Netherlands: Elsevier*.

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

Drira, A., Pierreval, H., & Hajri-Gabouj, S. (2006). Facility layout problems: A literature analysis. *IFAC Proceedings Volumes*, *39*(3), 389–400.

Duinkerken, M., & Atasoy, B. (2020). Quantative methods for logistics module 1.

Dumitrescu, I., & Boland, N. (2001). Algorithms for the weight constrained shortest path problem. *International Transactions in Operational Research*, *8*, 15–29.

Furmans, K., Schonung, F., & Gue, K. R. (2010). Plug-and-work material handling systems.

Glen, S. (2022). Traveling salesman problem and tsp art.

Gong, Y., & de Koster, R. (2011). A review on stochastic models and analysis of warehouse operations. *Logistics Research*, *3*(4), 191–205.

Gupta, S., & Jain, S. K. (2013). A literature review of lean manufacturing. *International Journal of Management Science and Engineering Management*, *8*(4), 241–249.

Gurobi. (2021). Managementpaper-4keyadvantagesofmovsheuristics.

Hamada, M., Martz, H. F., Berg, E. C., & Koehler, A. J. (2006). Optimizing the product-based availability of a buffered industrial process. *Reliability Engineering & System Safety*, *91*(9), 1039–1048.

Heskett, J. L. (1963). Cube-per-order index-a key to warehouse stock location. *Transportation and distribution Management*, *3*(1), 27–31.

Hoffman, K. L., Padberg, M., Rinaldi, G., et al. (2013). Traveling salesman problem. *Encyclopedia of operations research and management science*, *1*, 1573–1578.

IBM. (2022). Types of constraints. *IBM Integrated Analytics System*.

Janssen, J., Manca, R., & Volpe, E. (2013). *Mathematical finance: Deterministic and stochastic models*. John Wiley & Sons.

Jünger, M., Reinelt, G., & Rinaldi, G. (1995). The traveling salesman problem. *Handbooks in operations research and management science*, *7*, 225–330.

Karloff, H. (2008). *Linear programming*. Springer Science & Business Media.

Kay, M. G. (2012). Material handling equipment. *Fitts Dept. of Industrial and Systems Engineering North Carolina State University*, *65*.

Kenton, W., & James, M. (2021). Stochastic modeling.

Magzhan, K., & Jani, H. M. (2013). A review and evaluations of shortest path algorithms. *International journal of scientific & technology research*, *2*(6), 99–104.

Mahadevan, B., & Narendran, T. (1993). Buffer levels and choice of material handling device in flexible manufacturing systems. *European journal of operational research*, *69*(2), 166–176.

Marchet, G., Melacini, M., & Perotti, S. (2015). Investigating order picking system adoption: A case-study-based approach. *International Journal of Logistics Research and Applications*, *18*(1), 82–98.

Morse, P. M., Kimball, G. E., & Gass, S. I. (2003). *Methods of operations research*. Courier Corporation.

Mueller, M. (2020). Management of buffer systems in automotive stabilized production networks–a qualitative analysis. *Available at SSRN 3701532*.

Newell, C. (2013). *Applications of queueing theory* (Vol. 4). Springer Science & Business Media.

Nikolić, I. (2007). Total time minimizing transportation problem. *Yugoslav Journal of Operations Research*, *17*(1), 125–133.

Park, B.-C. (1987). Closest open location rule in as/rs. *Journal of Korean Institute of Industrial Engineers*, *13*(2), 87–95.

Pillac, V., Gendreau, M., Guéret, C., & Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, *225*(1), 1–11.

Quintanilla, S., Pérez, Á., Ballestín, F., & Lino, P. (2015). Heuristic algorithms for a storage location assignment problem in a chaotic warehouse. *Engineering Optimization*, *47*(10), 1405–1422.

Sargent, R. G. (2010). Verification and validation of simulation models. *Proceedings of the 2010 winter simulation conference*, 166–183.

Singh, S. K., & Yadav, S. P. (2016). A new approach for solving intuitionistic fuzzy transportation problem of type-2. *Annals of Operations Research*, *243*(1), 349–363.

Sniedovich, M. (1991). *Dynamic programming* (Vol. 297). CRC press.

Süße, M., & Putz, M. (2021). Generative design in factory layout planning. *Procedia CIRP*, *99*, 9–14.

Thacker, B. H., Doebling, S. W., Hemez, F. M., Anderson, M. C., Pepin, J. E., & Rodriguez, E. A. (2004). Concepts of model verification and validation.

Toth, P., & Vigo, D. (2002). An overview of vehicle routing problems. *The vehicle routing problem*, 1–26.

Umirzoqov, A. (2020). Using intermediate buffer temporary warehouses inside the working area of the gold mining quarry. *Scienceweb academic papers collection*.

Van Gils, T., Ramaekers, K., Caris, A., & De Koster, R. B. (2018). Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review. *European Journal of Operational Research*, *267*(1), 1–15.

Williams, A. (1963). A stochastic transportation problem. *Operations Research*, *11*(5), 759–770.

Yang, P., Yang, K., Qi, M., Miao, L., & Ye, B. (2017). Designing the optimal multi-deep as/rs storage rack under full turnover-based storage policy based on non-approximate speed model of s/r machine. *Transportation Research Part E: Logistics and Transportation Review*, *104*, 113–130.

Ye, Y., He, L., Wang, Z., Jones, D., Hollinger, G. A., Taylor, M. E., & Zhang, Q. (2018). Orchard manoeuvring strategy for a robotic bin-handling machine. *biosystems engineering*, *169*, 85–103.

Yu, Y., & De Koster, R. B. (2013). On the suboptimality of full turnover-based storage. *International Journal of Production Research*, *51*(6), 1635–1647.

Zhang, H., Ge, H., Yang, J., & Tong, Y. (2022). Review of vehicle routing problems: Models, classification and solving algorithms. *Archives of Computational Methods in Engineering*, *29*(1), 195–221.