Technische Universität München

TUM School of Management

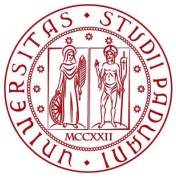
Innovative Optimization Models for the Part Feeding Problem in the Automotive Industry

Francesco Zangaro, Eng.

Vollständiger Abdruck der von der TUM School of Management der Technischen Universität München zur Erlangung des akademischen Grades eines Doktors der Wirtschafts- und Sozialwissenschaften (Dr. rer. pol.) genehmigten Dissertation.

Vorsitzende:		Prof. Dr. Gudrun Kiesmüller
Prüfer der Dissertation:	1.	Prof. Dr. Stefan Minner
	2.	Prof. Antonio Casimiro Caputo, Ph.D.

Die Dissertation wurde am 09.09.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Management am 15.10.2022 angenommen.



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

UNIVERSITY OF PADUA
DEPARTMENT OF MANAGEMENT AND ENGINEERING

PH.D. SCHOOL IN
MECHATRONICS AND PRODUCT INNOVATION ENGINEERING
Curriculum: Industrial Plants and Logistics
XXXIII CYCLE

Innovative Optimization Models for the Part Feeding Problem in the Automotive Industry

Ph.D. School Director:	Ch.ma Prof.ssa Daria Battini
Supervisor:	Ch.ma Prof.ssa Daria Battini
Ph.D. Candidate:	Francesco Zangaro, Eng.

Joint Ph.D. in collaboration with:
Technical University of Munich
TUM School of Management
Chair of Logistics and Supply Chain Management
Supervisor: Ch.mo Prof. Dr. Stefan Minner

TECHNISCHE
UNIVERSITÄT
MÜNCHEN

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Professor Stefan Minner and Professor Daria Battini, for their continuous support and very constructive criticism throughout my Ph.D. studies. I also want to thank them for encouraging and supporting me to spend 18 months of my Ph.D. program at TUM. I am very grateful that I was given the opportunity to be part of the research groups at the two universities.

Further, I am very grateful for the support of my former and current colleagues at the chair of Industrial Plant and Logistics at the University of Padua and the chair of Logistics and Supply Chain Management at the Technical University of Munich: Ph.D. Valentina Visentin, Ph.D. Serena Finco, Ph.D. Silvia Zuin, Riccardo Aldrighetti, Eleonora Florian, Dr. Christian Mandl, Dr. Szymon Albiński, Dr. Pirmin Fontaine, Dr. Miray Közen, Sebastian Malicki, Dr. Layla Martin, Santiago Nieto-Isaza, Dr. Partricia Rogetzer, Josef Svoboda, Dr. D.R.J. Dennis Prak, and Dr. Dariush Tavaghof Gigloo. Thank you for all the research discussions, for the after-work activities, and for all the times you have helped me. I would like to thank Evelyn Gemkow for her support and proofreading of my work.

Moreover, I would like to thank all the smart people I met during these five years at the University of Padua and the Technical University of Munich. I would like to thank Professor Monica Reggiani for the fruitful discussion on one of my papers, and I would like to thank the reviewers of this thesis and of the papers discussed in Chapter 5, Chapter 3, and Chapter 4 for their suggestions that helped improve them.

Finally, I would like to thank my family for their support and encouragement over the last few years.

Abstract

In an assembly line, assembly tasks are performed at the stations to assemble components on the product. At first, components need to be brought to the assembly stations. Components are usually stored in a warehouse and are delivered to the assembly stations by special vehicles that have limited space. In order to facilitate the delivery, special containers are used during the delivery operations. The Assembly Line Feeding Problem involves the selection of the most convenient container for each component in terms of costs and space for storage. Improving this activity translates to minimising the costs of operations that repeated over time would lead to considerable savings. The assembly tasks have precedence constraints among each other and need to be ordered while considering other constraints of the assembly line. This problem is known as the Assembly Line Balancing Problem. Improving these activities means reducing the working time, increasing productivity, and reducing costs. In order to study these activities, we investigated three major problems: the routing and scheduling decision for the delivery of components to the assembly line, the Assembly Line Feeding Problem, and combining the Assembly Line Feeding Problem with the Assembly Line Balancing Problem.

The first problem involves selecting a container for each component that is delivered to the assembly line. We implemented an optimisation model to solve the Assembly Line Feeding Problem, or line feeding mode selection. We applied an algorithm that provides a decision tree to explain the reasons for the line feeding mode selection. This decision tree identifies the attributes of the components that are most relevant for the selection and provides clear rules of thumb for the selection. To deal with the issue of infeasible instances that might occur due to the implementation of the decision tree, we developed a repair approach that ensures that the feasibility of the instances is restored and that the cost deviation from the optimal solution is minimal.

The second problem combines the decision of where the assembly tasks are performed in a multi-manned assembly line, also known as Assembly Line Balancing Problem, and the line feeding mode assigned to each component, also known as Assembly Line

Feeding Problem. We proposed an optimisation model and a heuristic algorithm to solve the combination of the two problems. We found that the combination of the Assembly Line Balancing Problem and the Assembly Line Feeding Problem can lead to a substantial cost reduction. We also performed a sensitivity analysis and find that the space at the Border of Line, the total volume of components delivered, and the total number of components delivered lead to an increase in the cost reduction due to the implementation of the combined approach. We also implemented the heuristic algorithm to a real application in the automotive sector and find that it leads to a cost reduction of 36.87%.

The last problem considers the delivery of the components to the assembly line performed by special vehicles. Congestion problems that occur in the delivery of components consist of a loss of resources and hinder other production activities. In order to deal with the congestion problems of these vehicles, we routed the vehicles and schedule the delivery of the components. We proposed an optimisation model and a heuristic algorithm to solve this problem. We compared three different facility layouts that are commonly found in manufacturing environments. For each of these layouts, we analysed the queue time and the congestion problems that the vehicles face. We found that the process layout is the one that is most likely to lead to congestion among the vehicles.

Keywords: part-feeding; manufacturing management; material handling; optimisation; heuristic;

Contents

List of Tables

List of Figures

List of Abbreviations

AGV	Automatic Guided Vehicles
ALBP	Assembly Line Balancing Problem
ALFP	Assembly Line Feeding Problem
ALNS	Adaptive Large Neighborhood Search
BoL	Border of Line
BOM	Bill Of Material
CART	Classification And Regression Tree
DSA	Direct Simulated Annealing
ICP	Inventory Control Problem
ILS	Iterated Local Search
IRP	Inventory Routing Problem
IRSP	Inventory Routing and Scheduling Problem
ISA	Indirect Simulated Annealing
JALBFP	Joint Assembly Line Balancing and Feeding Problem
KNN	k-Nearest Neighbor
KPI	Key Performance Indicator
LNS	Large Neighborhood Search
LPG	Liquid Petroleum Gas
MILP	Mixed Integer Linear Program(ming)
MIO	Mixed Integer Optimisation
MMALBP	Mixed Model Assembly Line Balancing Problem
NN	Neural Network
NP-hard	Non Polynomial-hard
OEM	Original Equipment Manufacturer
SGD	Stochastic Gradient Descent
SKU	Stock Keeping Unit
SSP	Supermarket Scheduling Problem
SVC	Support Vector Classification

SVM	Support Vector Machine
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

Chapter 1

Introduction

1.1 Motivation

The problems that are discussed in this thesis involve the assembly and production procedures of the automotive sectors. These problems are of great complexity because they involve many elements. In an assembly line, there is a large number of components that are delivered to the assembly line. In the most complex cases, there are more than 30 000 components are delivered to the assembly line. Around 15 000 components are delivered to a decentralised warehouse and from there they are delivered to the assembly line. A few hundred operators are involved in the delivery of the components from the warehouses to the assembly or production stations. In these stations, operators perform the productive or assembly operators. The assembly line is particularly long for vehicles that are produced in large volumes. i.e. non-premium vehicles. In those cases, the assembly line consists of more than 200 assembly stations where more than 900 operators perform the assembly operations. The whole assembly line has a daily production volume of about 250 vehicles (AUDI Brussels 2022). For premium vehicles, the assembly line is usually shorter with not more than 50 assembly stations where roughly 200 assembly operators work. The reason for this difference is that the production volume is lower for premium vehicles, i.e. a few dozen vehicles produced in a day. Regardless of the kind of vehicle produced, e.g. premium or non-premium, the activities performed are quite complex. Thus, it is important to improve the manufacturing procedures and achieve the highest efficiency in both the delivery of the components and the assembly operations of the vehicles.

In the past century, the automotive sector produced millions of cars for millions of families. Automotive vehicles that used to be a luxury at the beginning of the twentieth century are now a commodity that million of people own. The automotive sector strives

always to introduce innovative items in its products or to make niche optionals available to the masses, e.g. air conditioning and alloy rims, to meet the demand of customers and encourage sales. Non-premium automotive brands introduce innovative concepts in their manufacturing departments to provide products at affordable prices for customers. Since these customers are price sensitive and are demanding cutting-edge products, competition is very strong in the market. On the other hand, premium automotive brands still promote innovation but deal with small production volumes that limit the total profits that can be made. Although premium vehicles are sold at a high price, the quality of the products and the low volumes require the highest efficiency (Vitale and Giffi 2020a). Thus, all automotive companies strive very hard to achieve better performance, increase profits, reduce costs, and pursue innovative concepts in order to preserve or increase their market shares (Stellantis 2020; Volkswagen Group 2020). This can only be achieved if the most cost-efficient practices are implemented in a manufacturing environment so that unproductive activities and costly operations can be eliminated or at least minimised. A large amount of care is used to manage the activities that are performed in an assembly line since they are repeatedly performed and can greatly affect the price of the final product. The costs of a manufacturing environment can be greatly decreased if the manufacturing activities are properly managed (Wesselhöft 2020). Past research investigated the problems of a manufacturing environment. Nowadays, optimisation models and machine learning techniques can also be implemented to identify the best practices for the manufacturing environment. These techniques deal with some of the problems that managers must solve in real life.

As described in Farrell et al. (2003), the automotive industry incurs in high production costs compared to other environments, e.g. pharmaceutical and chemical. For instance, the pharmaceutical industry incurs in high costs for research and development to develop patents that last for years or decades for the production of medicine, medical treatments, or medical equipment. On the other hand, the automotive industry does not sustain these costs but rather deals with expensive production activities and high material costs. Farrell et al. (2003) discuss a cost breakdown for the production of automobiles. The production costs account for more than 65% of the total costs. These include material costs, manufacturing costs, maintenance costs, and transportation costs. Since the production costs have such a high percentage, reducing them is an effective way to increase profits. By contrast, improving research and development or warranty procedures of an automotive producer will only marginally affect the

final costs. Thus, improving manufacturing operations can lead automotive firms to realise high savings. One way to improve the manufacturing operations is to decrease transportation and production costs. If this is done well this can also decrease the cost of capital for the components since less stock needs to be stored in the manufacturing environments. These costs are part of the material costs. To reduce all these costs, a company needs to better organise its production activities so that they can be performed as efficiently as possible. If the production activities are carefully organised, unproductive operations are decreased and the costs can be minimised. Finally, a strong competitive advantage can be gained by an automotive firm if these costs are reduced.
In the past, technological advances were a constant factor for the automotive industry to the point that all automotive companies are used to them. Recently, customers are more concerned about the environment and are demanding more sustainable vehicles, i.e. a new generation of electric and hybrid vehicles (Anderson 2020). In the last ten years, this event brought up a big change in the whole industry which increased the complexity for producers (Freund et al. 2020). Manufacturers must produce a larger variety of vehicles compared to a decade ago. From a practical point of view, production environments that for a long time used to deal with few product models, i.e. petrol, diesel oil, and Liquid Petroleum Gas (LPG) vehicles, were forced to switch to large volumes of multiple variants of the same product model, i.e. regular, hybrid, mild-hybrid, natural gas, and electric vehicles. Each of these models requires different components and assembly operations (Elliott 2021; Küpper et al. 2020). For electric vehicles, gear boxes and transmissions need to be adapted, batteries are changing, and combustion engines are disappearing (Hehl et al. 2021). The increasing number of models leads to a higher number of components delivered. This larger number of components that are stored, delivered, and used in the assembly line leads also to higher costs for all the activities related to them. If these activities are not carefully managed, their costs could increase prohibitively. This leads to a higher level of complexity that the logistics departments must face.
These challenges affect not only automotive firms but also their suppliers, Original Equipment Manufacturers (OEMs), which must deliver a higher number of components with a higher level of technology or at a lower price. Although the problems, the issues, and the tools studied in this thesis focus on the automotive environment, they can also be applied to improve the activities in its satellite industry. Indeed, these two sectors deal with similar challenges and share common practices. Alternatively, other manufacturing environments, although very different from the ones of the automotive

industry, could also benefit from the implementation of the tools described in this thesis. For these reasons, the models and algorithms developed will be at first applied to an automotive environment, but will also be implemented in other industries similar to automotive. The approaches are generalised to include also other aspects and conditions that might occur in environments different from the automotive industry.

In recent years, there is also a new digitalisation trend that allows companies to improve their operations by reducing costs, increasing revenues, or improving profits. Traditional productivity levers are widely exhausted and companies turn to Industry 4.0 to achieve higher growth (Wee et al. 2015). Industry 4.0 consists of a set of technologies for the workplace. These technologies consist of data, connectivity, computational power, analytics, human-machine interactions, and digital-to-physical conversion. Due to competition, individual companies need to carefully analyse new opportunities and one of these opportunities is Industry 4.0. In 2015, Industry 4.0 was making up 19% of the total revenues of German companies Wee et al. (2015). For this reason and other similar situations, Industry 4.0 was in the past few years under scrutiny by companies as a way to achieve growth. Companies that employed these technologies, e.g. Amazon and Alibaba, were able to reach high values of revenues per employee (Simchi-Levi and Simchi-Levi 2020). Further, these technologies allow companies to deal efficiently with disruptions that are becoming more and more common (Van Alstyne and Parker 2021). However, these examples operate in the e-commerce industry and the manufacturing sector would like to reach the same results. To be able to achieve the same results, manufacturing companies need to translate the digitalisation trend into a concrete set of tools, applications, and guidelines. With Industry 4.0, there are two fields of study that can provide these tools, techniques, and methodologies: optimisation techniques and machine learning approaches. Companies are also struggling to find the right talent to develop and maintain these technologies (Jarvis 2020; Ramachandran and Watson 2021). This thesis aims also to provide tools and insights useful for managers for the implementation of Industry 4.0.

A first valid example is the containers that are used for delivering the components to the assembly stations. Components are delivered to the assembly stations in specific containers, also known as line feeding modes, to reduce the space required for storage at the Border of Line (BoL) and facilitate the delivery operations. The Assembly Line Feeding Problem (ALFP), also known as the part-feeding problem, involves the decision on the line feeding mode to deliver each component and assign a container to each component. This problem is usually solved with optimisation models that select

the best line feeding mode for each component (Caputo, Pelagagge, and Salini 2015; Limére, Van Landeghem, and Goetschalckx 2015). The computation times to solve an instance of the ALFP with one of these optimisation models is short. However, they lack clarity and do not provide a clear explanation for the line feeding mode selection of each component. They identify the line feeding mode that is economically most convenient to minimise the objective value. Therefore, managers would like to understand the reasons for the line feeding mode selection and do not have a general rule that can be applied in general so that they can autonomously perform such decisions, e.g. assign a line feeding mode to each component. Without these guidelines, managers and practitioners must rely daily on optimisation models for the ALFP. Relying on optimisation models in a daily routine is, although useful, impractical.

Another example is integrating assembly line balancing and feeding decisions in a single model. In a multi-manned assembly line, multiple workers perform assembly operations at the workplaces of the same station. The Assembly Line Balancing Problem (ALBP) involves the decision on how and in which order to perform the assembly operations in an assembly line. At a later step, the Assembly Line Feeding Problem (ALFP) is solved to select the line feeding mode of each component. When this occurs a high number of components must be delivered with special containers that allow their storage in a limited space. However, such containers require also costly additional operations performed by workers in the warehouse. The sequential implementation of the ALBP and the ALFP, that is common practice in an automotive environment, leads to the minimisation of the ALBP costs with a sequential increase of the costs related to the ALFP. Avoiding any consideration about the components while solving the ALBP could also lead to the need to perform rebalancing activities at a later point. On the other hand, the combination of these two problems can lead to savings in terms of costs. A few articles in the literature aim to solve the JALBFP (Battini et al. 2017; Sternatz 2015). However, these articles consider only the possibility to deliver the components directly from the warehouse or repacking them at the supermarket, a decentralised storage area. Further, these articles do not consider the costs associated with the delivery of components. Instead, they minimise the time of the activities or the number of operators. There is a need to understand the savings that can be achieved from implementing a joint model rather than a sequential one. To do this, a model that studies the ALFP and the ALBP with a higher level of detail is needed. This model should thoroughly consider the costs of the operations related to the ALFP and the ALBP. With this approach, we can achieve a full comparison between the integrated

and sequential models.

The last example is the issue of congestion problems that occur in the delivery of the components to the assembly line. The Inventory Routing Problem (IRP) selects the paths of the vehicles for the delivery of the components. Congestion problems occur in these activities and a multitude of logistics and productive environments and hinder production activities that are usually solved by the IRP (Bartlett et al. 2014; Roy, Gupta, and De Koster 2015). During the delivery of the components, whenever a vehicle meets another vehicle that is performing some delivery operations, it must wait in a queue before it can continue the roundtrip since overtaking is forbidden due to safety concerns. Not only do drivers spend time in queues, but also assembly operations at the stations could suffer. These congestion problems are unproductive activities that can disrupt the assembly operations performed in an assembly line. Congestion problems are known to reduce productivity also in other environments (Kammoun, Rezg, and Achour 2014). Other congestion problems that reduce the utilisation rate of vehicles are discussed in Zhang, Betta, and Nagi (2009). Chiew and Qin (2009), Fanti (2002), and Chiew (2012) investigate the problem of congestion for Automated Guided Vehicles (AGV) in a manufacturing environment. However, the congestion problems caused by regular vehicles are not investigated in the literature. Although the literature provides numerous articles that discuss the effects, the causes, and the solutions to congestion problems, the problem for the vehicles that travel in a manufacturing environment was always neglected.

These are all valid examples of problems that are thoroughly investigated in the literature. However, although the literature investigates some of the activities related to them, some assumptions, e.g. a consider a limited number of line feeding modes and only one operator per assembly station, prevent considering these problems. The current state of the literature does not provide information on the consequences of neglecting these problems. Investigating these problems and all related aspects will explain the conditions under which they negatively affect manufacturing activities. This investigation can explain the conditions when an alternative solution should be introduced to avoid additional costs. On the other hand, there is also the possibility that in certain environments and under certain conditions these problems will not affect the production activities and the regular solution available in the literature will be sufficient to solve these problems. Lately, with the course of digitalisation, optimisation models and machine learning algorithms became useful tools to investigate the activities that occur in a manufacturing environment. These tools can provide a valid solution for the

problems listed above so that the costs can be minimised while at the same time the constraints of the problems are respected. Furthermore, comparing the solutions to the new approach and the one usually implemented in a manufacturing environment can show the benefit that can be obtained.
The general purpose of this thesis is to provide models and algorithms for the ALBP and ALFP and consider also aspects that are neglected in the literature so that these approaches can be applied to manufacturing environments or industries different from the ones where these problems are most commonly encountered.
The specific objectives are:

1. develop optimisation models or heuristics that can provide exact and approximate solutions for the ALFP and ALBP;

2. apply these techniques to solve real-life instances that occur in a regular manufacturing environment;

3. generalise the problem and consider also those aspects of industries different from the automotive one where the ALFP and ALBP can be encountered;

4. investigate the conditions that make optimisation models and heuristic algorithms beneficial or necessary.

1.2 Research Questions

Based on the problems mentioned above, there is one research question for each problem that is investigated. Regarding the kind of containers used for the delivery of components to the assembly line, the research question is:

> What are the attributes of the components that are most relevant in the line feeding mode selection?

On the issues of selecting the right number of stations to perform the assembly activities in an assembly line:

> What is the most cost-efficient way to perform the assembly activities and deliver the components to a multi-manned assembly line?

Eventually, on the issue of congestion problems that occur during delivery of components to the assembly line:

How can we reduce the congestion in the part feeding process?

1.3 Outline of the Thesis

Figure 1.1 depicts the structure of the thesis. The outline can be divided into two major parts:

1. We provide a summary of the state of the art of the literature. In Chapter 2, we wish to discuss all the literature for the topics that are discussed in this thesis. The activities and operations that occur in a manufacturing environment are studied in the literature and optimisation models are provided to improve them. However, these models neglect factors or events that occur in other different manufacturing environments. This leads to problems, unforeseen situations, or additional costs that are not considered in the original approaches. We explain what is the current status of the literature. We identify the gaps in the literature that require further investigation or alternative solutions and explain how we aim to investigate these gaps. Eventually, these considerations on the state of the art of the literature are useful for the future research agenda.

2. Given the current literature, we implement and adapt existing methodologies to solve the problems that were ignored or consider aspects that were neglected also discussed in Section 1.1. Three manufacturing problems are discussed in the chapters:

    a) in Chapter 3, we improve the selection of containers for the delivery to the assembly line. Components that are stored in a decentralised warehouse must be delivered to the assembly line where space is scarce. It is important to assign a container, line feeding mode, to each component to reduce costs and ensure that these operations can be safely carried out. We provide an optimisation model with a cost objective function to solve this problem. Through the implementation of a machine learning algorithm, a decision tree can be obtained. The decision tree selects a line feeding mode for each component that needs to be delivered to the assembly stations. The decision tree provided is trained with synthetically generated data from 4 companies of different industries. The decision tree can clearly describe the attributes of the components for the line feeding mode selection, i.e. the assignment

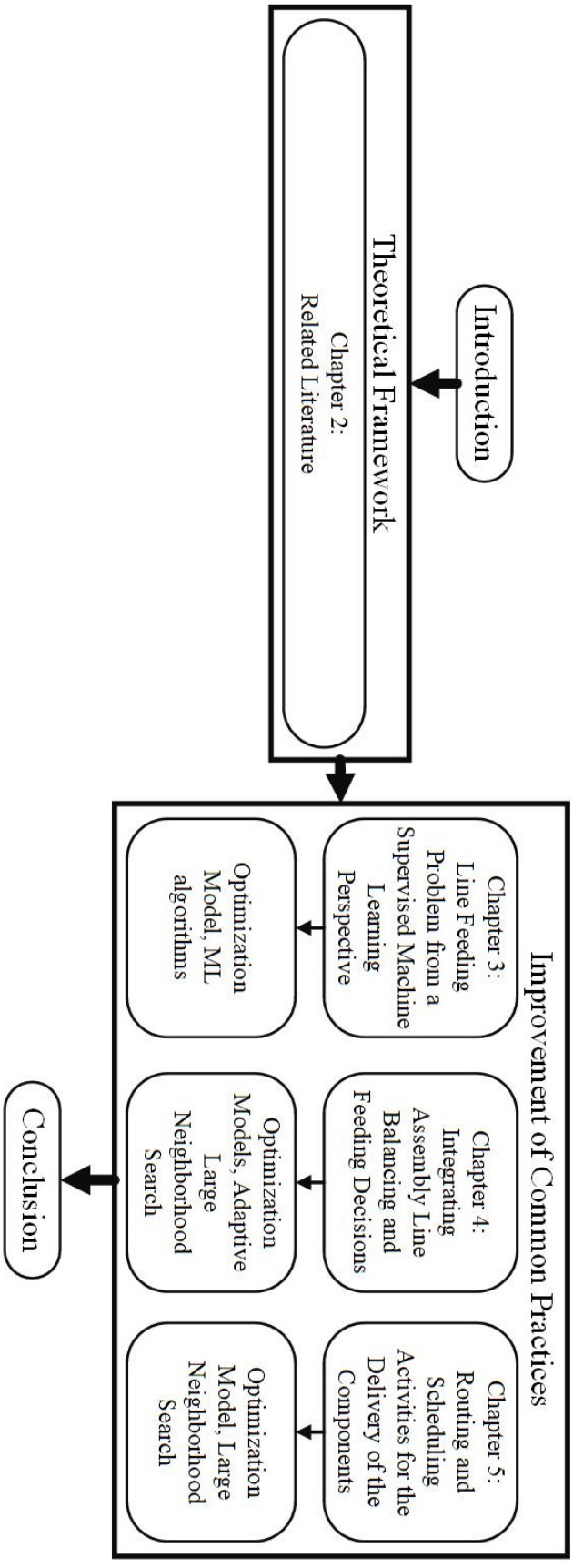


Figure 1.1: Outline of the thesis.

of a container to each component, so that managers can better understand which line feeding mode can be assigned to which component and why. This is useful when a new assembly line is implemented or re-balancing activities are performed;

b) in Chapter 4, we combine the decision on the tasks that should be performed at each station, ALBP, and the line feeding mode assigned to each component, ALFP. If these two problems are combinedly optimised, it is possible to achieve a larger amount of savings. The combination of these two problems is known as the Joint Assembly Line Balancing and Feeding Problem (JALBFP). We provide an optimisation model and a heuristic that can solve the JALBFP. We compare our model to a similar one found in the literature. We explain how the implementation of the JALBFP leads to a cost reduction compared to the sequential implementation of the ALBP and the ALFP. We analyse the cost reduction that can be achieved through the combined model. We perform a sensitivity analysis and explain that the savings increase when there is a higher number of components, components with a larger volume, and a shorter Length at the BoL. We also implement the heuristic in a real-life instance;

c) in Chapter 5, we improve the delivery of the components to a production area. Vehicles that are used for the delivery of the components might incur in congestion problems and queue times that can reduce the efficiency of the operations. We investigate how the routing and scheduling decisions of the vehicles can be combined in the Inventory Routing and Scheduling Problem (IRSP). We consider congestion problems that might occur in the delivery of components. To do so, we schedule the delivery of the components performed by the vehicles. We provide an optimisation model for the IRSP and a heuristic that can solve real-life instances. Both these approaches provide a timetable for the vehicles so that they can proceed to the delivery of the components and avoid waiting times. We investigate the three most common layouts of a production plant and find that the process layout is the one that leads to the highest level of congestion problems.

Eventually, we will provide a conclusion to this thesis. This conclusion will include the main contribution of the content of the thesis, the limitation of the methodology or the

analysis, and the future research agenda. The future research agenda points out the relevant questions that should be investigated.

1.4 List of the Articles

Three chapters of this thesis are related to three articles. These articles were either published in the proceedings of a conference or a peer-review journal:

1. Zangaro F., Battini D., 2018. "Integrating Routing and Scheduling Decisions for Enhancing the Part Feeding in an Automotive Environment", *Twentieth International Working Seminar on Production Economics*, Innsbruck.

2. Zangaro, F., S. Minner, and D. Battini. 2021. "A Supervised Machine Learning Approach for the Optimisation of the Assembly Line Feeding Mode Selection." *International Journal of Production Research*, 59 (16): 4881-4902.

3. Zangaro, F., S. Minner, and D. Battini. 2022. "The Multi-manned Joint Assembly Line Balancing and Feeding Problem." *International Journal of Production Research*. Advanced online publication. doi: 10.1080/00207543.2022.2103749.

Chapter 2

Related Literature

This thesis discusses topics that can be found in an automotive environment. Since these topics are connected, this chapter provides the whole literature review for the problems discussed in later chapters:

1. the application of machine learning techniques for classifying or identifying components in a manufacturing environment (Section 2.1 related to the topics discussed in Chapter 3);

2. the delivery of the components that are usually performed in an assembly line by solving the Assembly Line Feeding Problem (Section 2.2.1 related to the topics discussed in Chapter 3);

3. all the assembly activities that are performed in an assembly environment (Section 2.2.2 related to the topics discussed in Chapter 4);

4. the inventory routing problem implemented to improve the in-house transportation operations (Section 2.3 related to the topics discussed in Chapter 5).

2.1 Machine Learning for Classification Problems

Supervised Machine Learning is commonly applied to classification problems in the manufacturing environment (Pham and Afify 2005). Nakasuka and Taketoshi (1992) implement a machine learning model to develop decision trees that select the rule for scheduling the operations in the production line. By using computer simulations, the performance of the machine learning model is evaluated through computation experiments. Camci, Chinnam, and Ellis (2008) make use of alternative Support

Vector Machines (SVMs) model to monitor manufacturing processes that improve production quality. They discuss how the model can learn from out-of-control samples and it does not make any assumptions about the data distribution. Pinto, Luìs, and Moreira (2013) implement machine learning models such as K-Nearest Neighbor (KNN), Neural Networks (NNs), and SVMs to visually identify and recognize objects in a workplace for quality control purposes. Firstly, data is collected through a laser that provides the ranges and dimensions of the objects. Then, the machine learning models *classify* the objects based on the collected data, and their classification accuracy is compared. They also perform a comparative analysis of the machine learning algorithms to find the one with the best performance. Bertsimas and Dunn (2017) provide a Mixed Integer Optimisation (MIO) formulation to develop optimal decision trees. They compare this model with other heuristics and show that the new model outperforms the Classification And Regression Tree (CART). The main contribution of Chapter 3 is the assessment of the extent to which a machine learning approach can be used for developing a decision tree for the ALFP.

2.2 The Assembly Line Activities

This section provides the contributions concerning the ALFP, the ALBP, and the JALBFP.

2.2.1 Assembly Line Feeding Problem

As discussed in Battini et al. (2009), the ALFP is studied in the literature with an optimisation model since the first model is described in Bozer and Leon (1992). Battini, Boysen, and Emde (2013) introduce a classification framework for part supply in the automotive industry by considering traveling kitting, stationary kitting, and sequencing feeding modes, and Schmid and Limére (2019) classify the literature on the ALFP. For the sake of clarity, we are going to divide the articles that discuss the ALFP that include the sequencing line feeding mode and not. This line feeding mode consists of delivery different variants of the same component to the same station. In the order that is required for the assembly.

Assembly Line Feeding Problem without Sequencing

Caputo and Pelagagge (2011) introduce a descriptive model that designs the delivery system, describe the performance, and select the best line feeding mode between line stocking, kitting, and just-in-time delivery. They provide a model that can enhance the current performance of the line feeding activities and can be used to select the line feeding modes when planning a new facility before it might even be operational. They identify that the main disadvantage of kitting is the picking labour and line stocking policies require more space at the BoL for the storage of many components. Eventually, they also explain how the limited space at the BoL forces managers to rely on kitting as a line feeding mode rather than line stocking which is more cost-efficient. Limére et al. (2012) introduce a MILP that selects the best line feeding mode: kitting or line stocking. Their case study shows that a combination of kitting and line stocking ensures that the space at BoL is sufficient to store the components and avoid excessive costs. They identify key attributes of components that are usually kitted: components that would take up a large space at the BoL, small components, components that can be kitted in multiple parts per kit, and components that are in pallets. To improve this work by considering more than just constant travelled distances, Limére, Van Landeghem, and Goetschalckx (2015) present a new model that is also tested on a synthetic dataset. They explain how the kitting line feeding mode is assigned to a component not only based on the component's attribute but also on the attributes of the other components that are kitted. They call this phenomenon "free rider". The idea consists of the fact that the first kitted component includes also the costs for a kitted container. All the remaining components that are kitted in the same container will not have to bear this cost and, thus, are called "free riders". They explain that kitted components have usually a lower volume, a high number of variants, are supplied on pallets, have a low usage rate, and are usually batch picked. Caputo, Pelagagge, and Salini (2018) develop a cost model for the line feeding mode selection. In a sensitivity analysis, the relation between costs and volume per part and between weight and price per unit is analysed. This tells us that kitting and just-in-time delivery are appropriate for small containers containing few parts. Furthermore, they identify that labour costs and space occupation costs are the main factors driving the direct costs of line feeding policies. They suggest that the line feeding mode assignment depends also on economic issues such as the hourly wage, the daily demand, and the cost per square metre. Boudella, Sahin, and Dallery (2018) provide a mathematical model to assign Stock Keeping Units (SKUs) to either a robot or an operator. A case study from the automotive

sector is used to test the model. They find that component's attributes and the space floor can prevent the implementation of kitting automation. They identify the batch size as an element that can affect the kitting system cycle time. Zennaro et al. (2020) provide a mathematical model for the introduction of idle assembly islands in a manufacturing environment where the operators perform the part feeding activities in order to increase the production rate. Caputo, Pelagagge, and Salini (2021) provide a cost model to compare manual picking operations and automation-assisted parts retrieval for the kitting system. In order to test the cost function, a case study is provided to demonstrate its capabilities. A classification framework is introduced and a detailed planning and cost model is presented. However, they highlight how the approaches work with the assumption that the kitting line feeding mode is already assigned to the components.

Assembly Line Feeding Problem with Sequencing

Sali, Sahin, and Patchong (2015) provide a function that calculates the total costs of the three line feeding modes: line stocking, kitting, and sequencing. An analysis is performed to show which line feeding mode is convenient under what circumstances. They also notice the "free rider" phenomenon (see Limére, Van Landeghem, and Goetschalckx (2015)). Kitting is usually selected for small components with a lot of variants. As well, the usage rate of components increases their likelihood of being assigned to line stocking. Sali and Sahin (2016) use the previously developed cost model as an objective function of an optimisation model to find the best line feeding mode for each component and consider the constraints of the activities. They find that a cost reduction of about 10% can be achieved if an optimisation model is implemented. They also study the sensitivity of the costs regarding the kit container capacity and the space at the BoL. They find that these two factors significantly affect the costs. Baller et al. (2020) consider a model with nine different line feeding modes to reduce the costs of the line feeding activities and analyse a case study. In this case study, they implemented only some of the line feeding modes previously presented. They suggest that the ALBP and the ALFP should be integrated to reach significant cost reduction. As well, they highlight that flexibility of the length at the BoL can lead to a significant cost reduction for the ALFP. Zangaro, Minner, and Battini (2021) implement a MILP and use machine learning to train a decision tree for the line feeding mode selection. The decision tree identifies the features that are most likely to play an active role in the line feeding mode selection. They discuss the problems that can

arise from the implementation of a classification tool in the ALFP. Schmid, Limére, and Raa (2021) provide a mathematical model for the line feeding mode selection and to select the stations where the components should be stored. This model is tested by using artificially created data based on a case study with 3000 parts. They study how the line feeding problem can be solved for each component and different variants of the same component. They highlight that the variants of the same component should not be assigned to a different line feeding mode. This is because it is not very beneficial in terms of costs and it does not affect the line feeding mode selection but it increases the complexity of the problem. Additionally, they suggest that more space at the Border of Line (BoL) decreases the average costs. Müllerklein, Fontaine, and Ostermeier (2022) discuss a mathematical model for the line feeding mode selection and the production schedule. The study shows that the two problems combined lead to remarkable savings. The model is tested with a case study of a mixed-model assembly line. They suggest that the schedule type of the production activities should be taken into consideration when solving the ALFP. They explain that the joint scheduling and line feeding policy decision leads to only modest cost improvements compared to solving sequentially the two problems.

Table 2.1 provides a comparison of the various streams in the literature and Table 2.2 categorises the costs minimised in each article of the literature. For the sake of clarity, we are going to describe the cost categories. Since the articles use different names to describe the costs, we named the categories differently. These cost categories do not relate to one specific line feeding mode. For more details, please refer to Section 3.2 and 4.2.1. Seven cost categories are considered:

- *Replenishment costs* are incurred to perform operations that consist in replenishing the components in the decentralised storage area.

- *Preparation costs* consist of all the operations that take place in the decentralised storage area to prepare the components or *containers* for the delivery.

- *Transportation costs* are incurred during the transfer of the *containers* from the storage area to the assembly stations.

- *Picking costs* consist of costs for retrieving the components at the BoL.

- *Storage costs* consist of the costs incurred for stocking the components at the BoL or in the preparation area next to the storage area. Among the other costs, these include the costs for the surface.
- *Costs of capital* account for the costs of inventory holding capital to stock the components at the BoL.
- *Equipment costs* consist of the costs for the equipment used in the delivery.

Some of the articles discuss that these costs can be, in their case, neglected while others suggest that a multiplying factor could be added to the cost function to account for additional costs. For instance, one of the articles explains how the costs for the equipment include also its capital holding costs. However, we do not include these costs in our categorisation.
Some of the articles discussed in this section make some generalisations on the attributes of the components that are assigned to a line feeding mode. However, these assumptions are hard to visualise and explain. In Chapter 3, we provide an optimisation model for the ALFP.

2.2.2 Assembly Line Balancing Problem

The Assembly Line Balancing Problem (ALBP) is first described in Salveson (1955). Bowman (1960) provides a linear programming formulation for it. We refer the reader to Boysen, Fliedner, and Scholl (2008) for a complete understanding of the background and a description of the state of the art of the literature. The Simple Assembly Line Balancing Problem Type 1 (SALBP-1) is a specific case of the ALBP where, given a certain *cycle time*, either the number of the *stations* or the costs of the assembly activities are minimised. For the sake of clarity, we are going to divide the articles that deal with multi-manned assembly lines and not. The multi-manned assembly line is a special kind of assembly line where there are multiple operators at each station. These operators are performing different assembly operations in different parts of the product.

Table 2.1: Literature on ALFP

	Line Feeding Modes				Methodology		
	Line Stocking	Kitting	Sequencing	Externally Supplied Sequencing	Cost Function	Optimisation Model	Machine Learning
Caputo and Pelagagge (2011)	✓	✓	✗	✗	✓	✗	✗
Limére et al. (2012)	✓	✓	✗	✗	✓	✓	✗
Limére, Van Landeghem, and Goetschalckx (2015)	✓	✓	✗	✗	✓	✓	✗
Sali, Sahin, and Patchong (2015)	✓	✓	✓	✗	✓	✗	✗
Sali and Sahin (2016)	✓	✓	✓	✗	✓	✓	✗
Caputo, Pelagagge, and Salini (2018)	✓	✓	✗	✗	✓	✗	✗
Baller et al. (2020)	✓	✓	✓	✗	✓	✓	✗
Caputo, Pelagagge, and Salini (2021)	✗	✓	✗	✗	✓	✗	✗
Schmid, Limére, and Raa (2021)	✓	✓	✓	✗	✓	✓	✗
Müllerklein, Fontaine, and Ostermeier (2022)	✓	✓	✓	✗	✓	✓	✗
► Chapter 3	✓	✓	✓	✓	✓	✓	✓◄

Table 2.2: Cost categories of the literature on ALFP

	Replenishment	Preparation	Transportation	Picking	Storage	Capital	Equipment
Caputo and Pelagagge (2011)	✗	✓	✓	✗	✓	✓	✓
Limére et al. (2012)	✓	✓	✓	✓	✗	✗	✗
Limére, Van Landeghem, and Goetschalckx (2015)	✓	✓	✓	✓	✗	✗	✗
Sali, Sahin, and Patchong (2015)	✗	✓	✓	✓	✓	✗	✗
Sali and Sahin (2016)	✗	✓	✓	✓	✓	✗	✗
Caputo, Pelagagge, and Salini (2018)	✓	✓	✓	✓	✓	✓	✓
Baller et al. (2020)	✓	✓	✓	✓	✗	✗	✓
Schmid, Limére, and Raa (2021)	✓	✓	✓	✓	✗	✗	✗
Müllerklein, Fontaine, and Ostermeier (2022)	✓	✓	✓	✓	✓	✓	✗
▶ Chapter 3	✗	✓	✓	✓	✓	✓	✓◀

Regular Assembly Line Balancing Problem

Thomopoulos (1970) uses mixed model line balancing algorithms that help with loading the tasks to the *stations* more consistently. Scholl and Klein (1997) solve the SALBP-1 and introduced SALOME, a modification of the branch-and-bound algorithm that includes a new branching strategy and a bidirectional branching rule. This branch-and-bound algorithm outperforms other algorithms (FABLE and EUREKA). Boysen, Fliedner, and Scholl (2009) provide a modified approach for developing joint precedence graphs for the Mixed Model ALBP (MMALBP). They discuss how the probabilistic nature of the task duration is relevant to the resolution of the ALBP. Nourmohammadi and Eskandari (2017) propose a MILP for the ALBP and for the Supermarket Location Problem (SLP) which investigates the best location of the supermarket along to assembly line. This approach can be used sequentially to assign tasks to the assembly line and the location of the supermarket. The bi-level mathematical model can optimise the assembly line for the ALBP while taking into consideration the SLP.

Multi-manned Assembly Line Balancing Problem

Dimitriadis (2006) proposes a heuristic that is based on a Hoffmann procedure, i.e. an enumeration procedure to assign the tasks to the *workplaces*, for the multi-manned assembly line suggesting that if the product is large enough, multiple *workplaces* can be implemented in the same *station* to shorten the assembly line. Dimitriadis (2006) evaluate the approach provided by solving an instance of a real-world automotive assembly line. Sternatz (2014) proposes an enhanced multi-step bidirectional approach, called multi-Hoffmann heuristic, for multi-manned ALBP that considers multiple product models and *workplaces* per *station*. Sternatz (2014) shows that the enhanced multi-Hoffman heuristic is more efficient and effective compared to other heuristics such as the Avalanche algorithm. Roshani and Giglio (2016) provide a mathematical formulation and two Simulated Annealing heuristics for the multi-manned assembly line. They find that the Indirect and Direct Simulated Annealing (ISA and DSA) can find the optimal solution for the majority of small-sized problems. They implement the simulated annealing approach to solve a real-life instance of a Greek automotive manufacturer. They also use this approach to solve instances found in the literature. For small-size instances, the approach can find the optimal solution. For medium- and large-scale instances, it finds a solution with a shorter assembly line in 70% of the cases.

Giglio et al. (2017) propose a mathematical model for the multi-manned assembly line balancing problem with skilled workers that minimises the total operating costs of an assembly line. They compare the mathematical model to the one presented in Moon, Logendran, and Lee (2009) in the literature and they find that the total operating costs can be decreased. This is because their model can find better results in terms of computation times and the quality of the solution obtained. Martignago, Battaïa, and Battini (2017) present a model for balancing a multi-manned assembly line to minimise the total costs when different operator skills are involved at the same time and apply this model to a real industrial case. They argue that their model can be applied as a simulation tool to identify the best set of skills for training. Yilmaz (2020) proposes a Mixed Integer Linear Programming (MILP) for the ALBP with a bi-objective function that minimises the workload imbalance and the operational costs. This model determines the cycle time and the number of parts that must be delivered to an assembly line. Roshani et al. (2021) propose a MILP model that minimises the cycle time for the multi-sided assembly line balancing problem. They propose a heuristic to solve larger instances since the multi-sided Assembly Line Balancing Problem is NP-hard. They apply their model to an assembly line of a car body. They show that the proposed approach can solve middle- and large-scale instances more efficiently than other approaches.

2.2.3 Joint Assembly Line Balancing and Feeding Problem

Battini et al. (2016) provide a MILP model for the JALBFP by minimising total annual manufacturing costs. The authors consider only line stocking and kitting and they did not solve a multi-manned problem. Battini et al. (2017) present a model for the JALBFP that minimises the total number of workers by smoothing the ergonomic workload between them. The authors do not consider multi-manned assembly lines and costs linked to space and transportation. They discuss how the joint model decreases the space needed for the storage of the components at the BoL. They also argue that the production time decreases while the area of the supermarket increases. They suggest that the idle time is lower in the solution of the joint model compared to the sequential one, though they mention that the idle time might be a misleading KPI for the problem. Sternatz (2015) extends Sternatz (2014) by providing an improved version of the multi-Hoffmann approach for the multi-manned JALBFP. This model minimises the total working time per order. However, it does not consider an assignment of different line feeding modes to the components. In the

computation experiment, 175 instances of 50 and 20 tasks of the data set provided by Otto, Otto, and Scholl (2013) are solved. Sternatz (2015) explains how potential savings can be achieved if the ALFP and the ALBP are solved jointly rather than sequentially. Sternatz (2015) identifies that an average total cost reduction of 5% can be achieved through the joint model compared to the sequential approach. It also shows how the proposed model leads to a reduction in production time compared to the model presented in Sternatz (2014). Wijnant, Schmid, and Limére (2018) explain that the ALBP and the ALFP are interrelated. Indeed, they indicate that the decision of the line feeding mode assigned to each component depends on which *station* this component is needed and the tasks that are performed there. Calzavara et al. (2021) provide a mathematical model for a regular assembly line with the assumption that a task requires one component that can be delivered with two line feeding modes: line stocking or kitting. The objective function minimises the costs of the assembly and line feeding activities. They explain how the joint mode provided leads to a cost reduction compared to the sequential approach. In the case studies, the cost reduction is due to a more efficient space utilisation in the assembly line with a lower number of assembly stations implemented. Zangaro. Minner, and Battini (2022) provide an optimisation model and a heuristic to solve the multi-manned joint assembly line balancing and feeding problem. To show how this model can be implemented in a real-world scenario, a real instance is solved. They show that the costs for the line feeding activities are higher when a sequential model is implemented. If these costs are considered in a joint model, cost savings can be achieved. Table 2.3 provides a comparison of the various studies discussed above, wherein we categorised them for the three problems (ALBP, ALFP, and JALBFP), whether the objective function minimises costs, if they provide an optimisation model, and include other details of the models. Table 2.4 provides the cost categories and the line feeding modes of the articles that solve the multi-manned JALBFP. Differently from Table 2.2, we consider also the assembly costs which include the costs to perform the assembly tasks in the stations. While the ALBP is broadly studied in the literature, the JALBFP is less investigated. All the more the JALBFP is not studied with a high level of detail. The main contribution of Chapter 4 is the assessment of a joint model for the ALBP and the ALFP.

Table 2.3: Comparison of available literature on the ALBP, ALFP, and JALBFP.

	Problems			Methodology		Level of detail	
	ALBP	ALFP	JALBFP	Cost reduction	Optimisation model	Multi-manned	Kitting and sequencing
Scholl and Klein (1997)	✓	✗	✗	✗	✓	✗	✗
Dimitriadis (2006)	✓	✗	✗	✗	✗	✓	✗
Sternatz (2014)	✓	✗	✗	✓	✗	✓	✗
Roshani and Giglio (2016)	✓	✗	✗	✗	✗	✓	✗
Limére, Van Landeghem, and Goetschalckx (2015)	✗	✓	✗	✓	✓	✗	✗
Sali, Sahin, and Patchong (2015)	✗	✓	✗	✓	✗	✗	✓
Sali and Sahin (2016)	✗	✓	✗	✓	✓	✗	✓
Caputo, Pelagagge, and Salini (2018)	✗	✓	✗	✓	✓	✗	✗
Zangaro, Minner, and Battini (2021)	✗	✓	✗	✓	✓	✗	✓
Sternatz (2015)	✗	✗	✓	✗	✓	✓	✗
Battini et al. (2017)	✗	✗	✓	✗	✓	✗	✗
Calzavara et al. (2021)	✗	✗	✓	✓	✓	✗	✗
▶ Chapter 4	✗	✗	✓	✓	✓	✓	✓▲

Table 2.4: Cost categories and line feeding modes of available literature on the JALBFP.

	Sternatz (2015)	Battini et al. (2017)	Calzavara et al. (2021)	▶ Chapter 4 ◀
Cost categories				
Assembly	✓⋆	✓⋆	✓	✓
Replenishment	✗	✗	✗	✓
Preparation	✓	✓	✓	✓
Transportation	✓	✗	✓	✓
Picking	✗	✗	✗	✓
Equipment	✗	✗	✗	✓
Line feeding modes				
Line stocking	✓	✓	✓	✓
Kitting	✓	✓	✓	✓
Sequencing	✗	✗	✗	✓

⋆ Sternatz (2015) minimises the total production time and Battini et al. (2017) minimises the number of required assembly and supermarket operators both these measures can be seen as costs

2.3 Inventory Routing Problem

We divide the literature review into a taxonomy section and a section about the articles that investigate the IRP.

2.3.1 Taxonomy on Traveling Problems

The Traveling Salesman Problem (TSP) studies how a vehicle can travel through the cities in order to minimise the route without taking into consideration the capacity of the vehicles. This problem is first described by (Robinson 1949). The (VRP) addresses how goods can be delivered to the stations in order to minimise the delivery time and considers the capacity of the vehicles. This problem is first described in Dantzig and Ramser (1959) and inherits the Non Polynomial-hard (NP-hard) complexity of the TSP. The IRP, as introduced by Bell et al. (1983), routes vehicles through the stations by optimizing the part-feeding process. As described in Abdul Rahim et al. (2014), the IRP is a combination of the Inventory Control Problem (ICP), which involves the decision on the amount of components stored at the station, and the VRP, which consists of developing routes for the delivery of components, also known as references. The IRP plans the delivery of the references from a warehouse, minimises the stocks and the delivery time, and considers the roundtrip time when calculating the quantity that is delivered to the stations. Indeed, the IRP schedules the delivery to the stations to calculate the quantity of the references that must be delivered. The IRP inherits the NP-hard complexity from the VRP. Periodic IRPs schedule the delivery of the references that is performed repeatedly over a certain time period (Aksen et al. 2014). The Supermarket Scheduling Problems (SSP) consist of scheduling the delivery operations correctly from the supermarket to the assembly stations (Emde and Boysen 2011). In other words, the trips of the tow trains are scheduled to minimise the stock at the stations and reduce the number of delivery tours. In Chapter 5, we wish to solve a new problem that considers the delivery of references, usually part of the IRP, and the scheduling of the delivery, usually part of the SSP. However, combining the IRP and the SSP will lead to a new kind of problem that has not yet been considered. While the IRP routes the tow trains to the stations where they perform part-feeding operations, the SSP schedules such procedures to minimise the stock at the stations. The IRSP encompasses the IRP and the SSP and develops schedules and routes for the delivery of the references. Congestion problems

are not solely related to the routes and the schedules of the delivery, but both these factors combinedly. Implementing a regular two-index formulation of the IRP will not allow tracking the congestion problems that might occur during the delivery. If we wish to study congestion problems, we need to combine the route selection and the schedule of the vehicles. For this reason, we need to define a new and more complex problem and rely on a more complex formulation than for usual IRPs. The IRSP inherits the NP-hard complexity from the IRP.

2.3.2 Inventory Routing Problem

The IRP is studied in the literature from different perspectives. Campbell and Savelsbergh (2004) develop a two-step approach. Firstly, they develop an integer programming approach for the scheduling, then, based on a heuristic, the routing is adjusted. Coelho and Laporte (2013) provide a branch-and-cut formulation for the IRP with multiple products and multiple vehicles that leads to exact solutions. They evaluate the model proposed by introducing a set of benchmarking instances. Andersson, Christiansen, and Desaulniers (2015) provide a decomposition approach for the IRP that involves the distribution of liquid natural gas. This decomposition approach generates the paths that are then solved with a branch-and-bound algorithm. The periodic IRP develops a schedule for the delivery which is repeated identically over time. In the numerical results, the new formulation is able to solve instances obtained from the literature with lower computation times. Gaur and Fisher (2004) provide a three-step approach to optimise the periodic delivery of references from the warehouses to the stores of a supermarket chain. In this case, the scheduling problem is solved sequentially after solving the routing problem. They discuss the implementation of the approach in a supermarket chain, the challenges it faced, and the benefits that it experienced. Aksen et al. (2012) develop two formulations to solve a periodic IRP. This heuristic is used for routing the collection of waste vegetable oil in order to minimise the total costs. These approaches are applied for instances with up to 30 nodes. In their real-world implementation, they find that vehicle costs account for the most to the objective function. They also argue that increasing the vehicle capacity decreases the most the objective function. In order to solve larger instances of the same problem, Aksen et al. (2014) present an Adaptive Large Neighbourhood Search. They compare the Adaptive Large Neighbourhood Search (ALNS) that they propose to a Mixed Integer Linear Programming (MILP) model and find that the

ALNS outperforms the MILP for large-scale instances in terms of computation times and solution quality. Bard and Nananukul (2009) develop a two-step approach for a multiperiod IRP in a manufacturing supply chain that first solves the ICP by setting the delivery quantity and then considers the VRP. They compare the computation times of the proposed approach to the ones of other approaches found in the literature. Fransen et al. (2020) solve with an extensive discrete-event simulation the routing problem for Automatic Guided Vehicles (AGV) in a manufacturing environment. This model aims to intervene when the deadlock is irreversible by rerouting the AGVs that are waiting in the deadlock. Chiew and Qin (2009) present an algorithm that adapts a bitonic merge sort algorithm for the routing and scheduling problem to avoid deadlocks when AGVs are implemented. They compare their approach to the ones found in the literature and discuss the possible applications. However, these papers focus on deadlock issues that occur with AGVs when they turn, the problem of the delivery with tow trains is different since it considers the congestion problem that occurs during the delivery of the components with tow trains. Indeed, deadlocks between AGVs might require, in the most extreme cases, human intervention to be solved since the operator has to manually reroute the vehicles. When solving the problem of deadlocks that can arise for AGV, the aim of the model provided by Chiew and Qin (2009) is to achieve no deadlock at all since even one deadlock could result in the loss of a significant amount of money. This problem is different from the one described in Chapter 5.
Although all these articles study the IRP, the congestion problems that arise for the delivery of components in a manufacturing plant are not investigated in the literature. Chapter 5 aims to fill the gap in the literature by providing a mathematical formulation for the periodic IRSP. Through this model, we identify the degree of importance of congestion in the part-feeding problem.

Chapter 3

Assembly Line Feeding Problem from a Supervised Machine Learning Perspective

Based on

Zangaro, F., S. Minner, and D. Battini. 2021. "A supervised machine learning approach for the optimisation of the assembly line feeding mode selection." *International Journal of Production Research*, 59 (16): 4881-4902.

The Assembly Line Feeding Problem (ALFP) involves the delivery of components to the production area. Previous models minimse the delivery costs and optimally assign each component to a line feeding mode between line stocking, kitting, and sequencing but cannot provide easily comprehensible guidelines. We use the Classification And Regression Tree (CART) algorithm to develop, in a supervised way, a decision tree based on problems that are solved with a Mixed Integer Linear Programming (MILP) model for training purposes. Based on selected attributes of the components and the manufacturing environment, the decision tree suggests a line feeding mode for every component. For a synthetically determined training and evaluation data set, we find that the classification tree can predict the line feeding mode with an average classification accuracy of 78.49%. After the decision tree is implemented and a line feeding mode is selected for each component, an infeasible solution might occur. We develop a repair approach that solves this problem with an average cost deviation from the optimal solution of 0.38%.

3.1 Introduction and Background

In a manufacturing environment, several production and assembly activities are performed. After the production, the components are stored in warehouses, and line feeding activities involve the delivery of these components to the assembly line. The Assembly Line Feeding Problem (ALFP) is a classification problem where a line feeding mode is assigned to each component (Battini et al. 2015). There are many optimisation and heuristics that minimse the costs of the part feeding activities for each component by comparing the modes or selecting the most appropriate one, e.g. Caputo, Pelagagge, and Salini (2018), Limére, Van Landeghem, and Goetschalckx (2015). However, these approaches are complicated, and they do not provide an understanding of how the results were obtained.

Practitioners want to understand the guidelines that help with the selection of the optimal line feeding model for new components in a supermarket. They need to understand the attributes of the components that drive the selection. Furthermore, suppliers can change the size, kind, and capacity of boxes on a daily basis. Operators continue to deliver these components in the line feeding mode they originally selected, since optimisation models are impractical for daily utilisation and there are no other accessible decision support systems. Consequently, there is the need to develop a method that provides easy-to-use guidelines for practitioners. Decision trees are a powerful tool in these scenarios.

Our model enhances and extends an optimisation model adapted from Sali and Sahin (2016) and the objective function previously presented in Sali, Sahin, and Patchong (2015). In contrast to the original model, we consider *stationary kits* instead of the *traveling kit* since the real data used in this work was provided by companies that mostly implement *stationary kits*. We consider *externally supplied sequencing* as an additional line feeding mode that was previously neglected, before introducing some constraints to account for its implementation. Further, we consider the cost of capital and the equipment costs that affect the line feeding activities.

Supervised machine learning algorithms and models proved to be very effective in classification problems. We apply a supervised machine learning algorithm to the ALFP. Since the model is supervised, it needs to learn from instances that are solved with the optimisation model. Unlike previous machine learning algorithms that are *trained* with *features* and *classes* of problems already available in the literature, our model uses the results of an optimisation model for *training*, *validating*, and *testing*.

An approach for developing decision trees is used and parameterized.
If this decision tree is used and a line feeding mode is selected for each component, there is the possibility that an entire instance is infeasible due to joint capacity constraints. Therefore, we develop a repair approach that assigns a line feeding mode to certain components until a feasible solution is reached. To validate this method, we define four main Key Performance Indicators (KPIs): classification accuracy, the confusion matrix, the number of infeasible instances before the repair approach, and the total cost deviation after the repair approach. The decision tree has a classification accuracy of 78.49%. The repair approach can decrease the average cost deviation from the optimal solution to 0.38%. Since machine learning processes are affected by the quality of the data, our model is tested with data synthetically generated from data sets that were collected from four companies: an automotive manufacturer, an Original Equipment Manufacturer (OEM), a producer of electrical motor, and a producer of naval equipment.
The remainder of this chapter is structured as follows: Section 3.2 describes the problem and the general assumptions of the model, Section 3.3 provides the optimisation model, the machine learning algorithm, and the repair approach, Section 4.4 describes our KPIs, the data generation process, the results, and an exemplary application of the decision tree, and Section 6 provides a summary and ideas for future research.

3.2 Problem Description and Assumptions

Components, also known as references, are usually stored in a special warehouse that is known as supermarket and located near the assembly area. For each component, there are several different variants that consist of similar objects. They might differ in one or more attributes i.e. colour, texture, or material. For instance, a case that is installed on an electronic appliance might come in different colours and materials. During each assembly operation, only one variant is assembled to the product. Besides, every component is delivered to only one assembly station.
All the wagons of the tugger train have an identical maximum capacity in terms of volume (m^3) that limits the amount of components that can be transported and delivered.
There are four commonly implemented line feeding modes:

- *Line Stocking* employs a kanban call-sign or a consumption renewal method.

Usually, each *box* contains multiple identical parts of the same object. Since each component has its individual consumption rate, the number of *boxes* may vary. According to Sali and Sahin (2016), the operator can load or unload several *boxes* of the same component at once whenever this is possible. Components must be identified during the picking operations.

- *Stationary kit* refers to the collection of the components needed to perform the assembly operations of a single assembly station. This approach consists of collecting different components from different repositories and placing them together in a single *kit container*. As opposed to the *traveling kit*, which moves along the assembly line, the *stationary kit* is delivered and remains at the BoL of a station. If the weight and volume allow it, multiple components are simultaneously retrieved by the picking operator.

- *Sequencing* means that different variants of the same component are collected and placed inside a *container* in the same order as they will be consumed in the assembly line. Each component has its own *container* where its variants are placed in order. *Sequenced parts* are stored at the BoL.

- *Externally supplied sequencing* is similar to *sequencing*. All the different variants of the same component are placed in the same order as they are needed in one *container*. In contrast to *sequencing*, the preparation activities are not performed at the production plant, but by the supplier. This happens if the component has a high number of variants or if the component's box is larger than a certain size and you want to limit the total supermarket space.

Generally, components are placed into *containers* for all the four line feeding modes. More specifically, for *line stocking*, *stationary kitting*, and *sequencing* (*regular* or *externally supplied*), the objects to be processed are respectively *boxes*, *kit containers*, and *sequenced parts* or *parts*. Figure 4.1 depicts the supermarket and the BoL.
We assume that the operations are subdivided into six main cost elements:

- *Preparation before assembly* is an operation that takes place in the supermarket. The operator collects the components from the shelves and, if and where necessary, loads them onto the appropriate *container*.
  Since each component has its individual consumption rate, the number of *containers* may vary between components. The number of *kits* and *sequenced*

parts that are prepared between two deliveries is equal to the number of takts, which is the number of products assembled in the same time interval. The operator can load or unload several *containers* of the same component at once whenever this is possible.

- *Transportation* involves the transfer of the *containers* that are already on the tow train from the supermarket to the assembly line. In a milk-run, the tow trains travel through the whole assembly line, regardless of the number of stations. *Kitted parts* are placed on a cart that is attached to the tow train before departure. We also assume that the traveling time is constant and not affected by delays and queues.

- *Picking at the BoL* consists of retrieving the components at the BoL. *Line stocked* components need to be identified before they can be picked at the BoL. *Kitted* and *sequenced parts* do not require identification during the picking operations as these line feeding modes are designed to be easily recognized. *Containers* are stored at the BoL, where they can easily be retrieved by the assembly operators.

- *Storage* consists of stocking the components at the BoL or in the preparation area next to the supermarket. There is an area near the supermarket where the *kitting* and *sequencing* operations are performed. Furthermore, *kit containers* and *sequenced parts* are stored in a buffer area near the supermarket before they are attached to or loaded onto the tow trains.

- *Cost of capital* account for the costs of inventory holding capital for a stock of the components at the BoL.

- *Equipment costs* consist of the costs for all the *containers* that are used in the line feeding activities.

Table 3.1: Sets, indicis, parameters, and attributes of the component used in the model

Sets and Indicis of the model		
Not.	Definition	Range
$\|S_k\|$	Number of variants of a component k	—
i	Variant index	$i \in I$

I	Set of variants of components	—
k	Component index	$k \in K$
K	Set of components to deliver	—
n	Station index	$n \in N$
N	Set of stations	—
S_n	Set of components assembled in a station n	—
S_k	Set of variants of a component k	—
u	Line feeding mode index	$u \in U$
U	Set of line feeding modes	—

Parameters of the model

Not.	Definition	Unit
ζ_n	Available length at workstations	m
ξ	Available depth at BoL	m
ψ	Preparation and delivery batch size	pcs
ω	Maximum volume occupied by a component with multiple variants in the supermarket	m^3
A^{sta}	Depth of a stationary kit container	m
B^{sta}	Length of a stationary kit container	m
C_0	Labour cost per time unit	$\frac{€}{s}$
C_{esc}	Purchasing costs for preparation activities externally performed	€
C_{kit}	Daily cost of the kit cart	$\frac{€}{day}$
C_{m^2}	Periodic rental cost per square meter	$\frac{€}{day\ m^2}$
C_{sh}	Daily cost of the shelf	$\frac{€}{m^3 day}$
C_{sq}	Daily cost of the sequenced cart	$\frac{€}{day}$
C_v	Daily vehicle unit cost	$\frac{€}{day\ vehicle}$
d	Interspace between two consecutive end products on the assembly line	m
D	Distance travelled by the tow train in one roundtrip	m
j	Daily interest rate	%
L	Number of levels of the shelves used for line stocked components at the BoL	—
L_{seq}	Maximum number of variants for the sequenced kit	pcs
M^{sta}	Weight capacity of a stationary kit container	kg
v_0	Speed of an operator	$\frac{m}{s}$

V	Production volume in the production period	pcs
Vol^{sta}	Volume capacity of a stationary kit container	m^3
v_t	Speed of the tugger train	$\frac{m}{s}$
T	Production period	s
Y	Capacity of the tugger train	m^3
	Time needed for	
t_{gl}	A single movement of grasping boxes and loading them.	s
t_{gsl}	A single movement of grasping line stocked parts for assembly.	s
t_{gul}	A single movement of grasping boxes and unloading them.	s
t_{il}	A single operation of identification.	s
t_{lk}	A single movement of loading a kit container on the train.	s
t_{pak}	A single movement of picking during the assembly in kitting mode.	s
t_{pas}	A single movement of picking during the assembly for the sequencing mode.	s
t_{pk}	A single movement of picking variants during the preparation in a kitting mode.	s
t_{ps}	A single movement of picking and loading variants during the preparation in a sequencing mode.	s
t_{uk}	A single movement of unloading a kit container from the train.	s
t_{us}	A single movement of unloading the sequenced parts at BoL.	s
Attributes of the components		
Not.	Definition	Unit
θ'_k	Average number of sequenced parts of the variant i of component k unloaded at once	pcs
θ'_{ki}	Average number of boxes of the variant i of component k handled at once	boxes
θ_k	Number of parts of the component k picked at once during the assembly	pcs
θ_{ki}	Average number of parts of the variant i of component k picked at once during the preparation	pcs
τ_{ki}	Usage rate of the variant i of component k	$\frac{pcs}{s}$

a'_k	Maximum number of boxes of the component k that can be handled at once	boxes
a_k	Maximum number of parts of the component k that can be handled at once	pcs
A'_k	*Depth of a box of a component k*	m
A_k	*Depth of a part of a component k*	m
B'_k	*Length of a box of a component k*	m
B_k	*Length of a part of a component k*	m
c_k	Bill of Material (BOM) coefficient of a component k	pcs
h_k	Price of one single part of component k	€
M'_k	*Weight of a box of the component k*	kg
M_k	*Weight of a part of the component k*	kg
p_k	*Number of parts per box of a component k*	pcs
r_{ki}	Number of boxes of the variant i of the component k consumed between two successive deliveries	boxes
Vol'_k	*Volume of a box of the component k*	m^3
Vol_k	*Volume of a part of the component k*	m^3

The attributes written in *italics* are collected from company records. The attributes underlined refer to the dimensions of 260 boxes listed in Hemminki, Leipala, and Nevalainen (1998), which we assign if the dimensions of the boxes were missing in the original record provided by the company based on their volumes. Each parameter

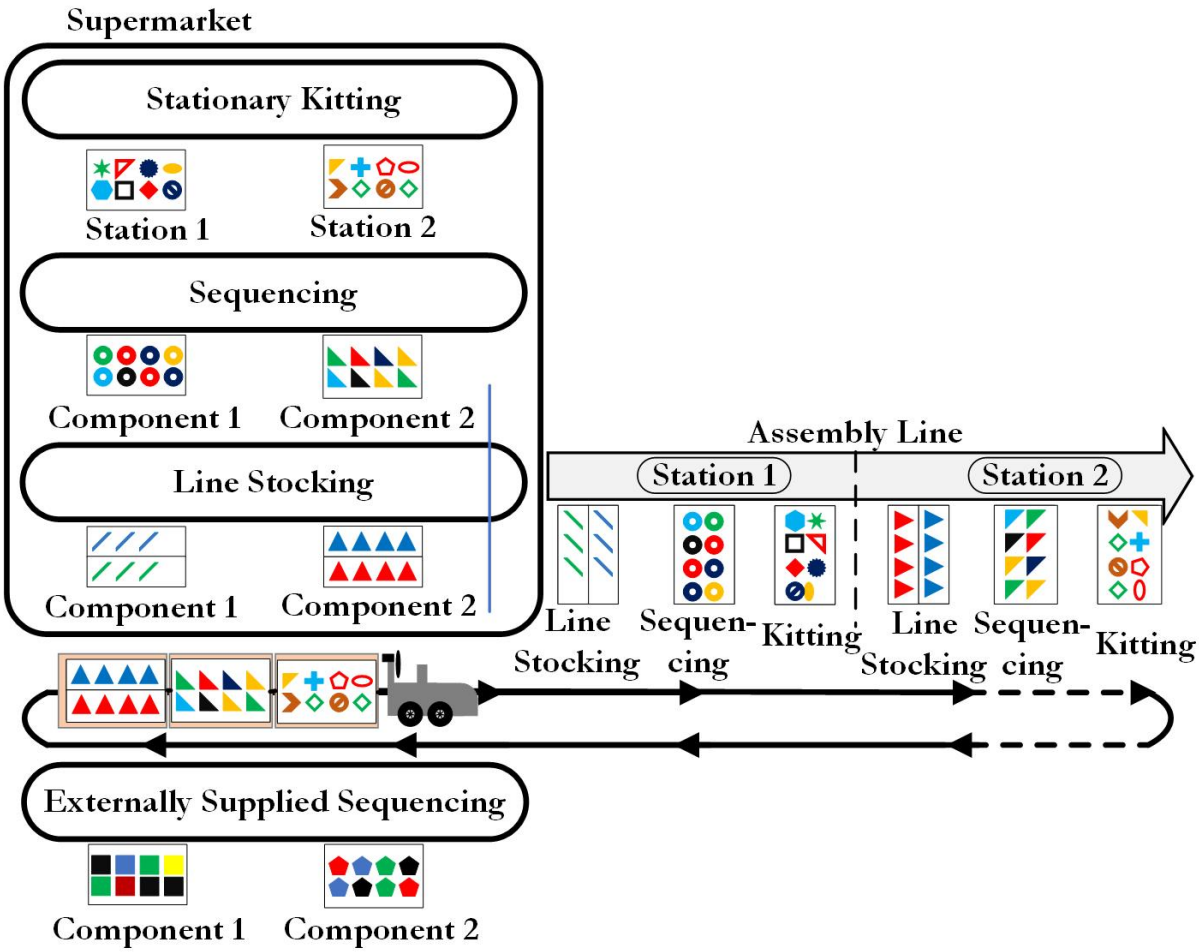


Figure 3.1: Supermarket area and assembly line

or attribute is provided with an accurate value, a list, or a range. The lists are provided in curly brackets. For the ranges, we provide the minimum and maximum value in squared brackets. For the attributes that are calculated, the formulas are provided and a detailed description is presented in Sali and Sahin (2016). The parameters are taken from Sali and Sahin (2016), Limére, Van Landeghem, and Goetschalckx (2015), and Caputo, Pelagagge, and Salini (2018).

Table 3.2: Data and ranges of the sets, parameters, and attributes of the components

Notation	List, Range, Value, or Formula	Unit
	Sets	
$\|S_k\|$	$[1, 5]$	—
I	5	—

K	{100;150;200}	—
N	{10;15;20}	—
	Parameters of the Model	
ζ_n	$[8, 10]$	m
ξ	$[3, 5]$	m
ψ	1	pcs
ω	1	m^3
A^{sta}	$[0.4, 1]$	m
B^{sta}	$[0.3, 0.8]$	m
C_0	0.0083	$\frac{€}{\mathrm{s}}$
C_{esc}	125	€
C_{kit}	$[0.1, 0.4]$	$\frac{€}{\mathrm{day}}$
C_{m^2}	0.2	$\frac{€}{\mathrm{day\,m^2}}$
C_{sh}	0.075	$\frac{€}{\mathrm{m^3 day}}$
C_{sq}	$[0.05, 0.2]$	$\frac{€}{\mathrm{day}}$
C_v	22	$\frac{€}{\mathrm{day\ vehicle}}$
d	1.5	m
D	$[200, 400]$	m
j	0.0137	%
L	3	—
L_{seq}	5	pcs
M^{sta}	$[30, 60]$	kg
t_{gl}	$[1.4, 2.1]$	s
t_{gsl}	$[1.384, 2.076]$	s
t_{gul}	$[1.4, 2.1]$	s
t_{il}	$[0.4, 0.6]$	s
t_{lk}	$[1.04, 1.56]$	s
t_{pak}	$[0.88, 1.32]$	s
t_{pas}	$[1.04, 1.56]$	s
t_{pk}	$[1.52, 2.28]$	s
t_{ps}	$[1.12, 1.68]$	s
t_{uk}	$[1.04, 1.56]$	s
t_{us}	$[1.12, 1.68]$	s
T	28800	s
v_0	1	$\frac{\mathrm{m}}{\mathrm{s}}$

V	$\{16; 32; 36\}$	pcs
Vol^{sta}	$[0.2, 0.7]$	m^3
v_t	1.38	$\frac{m}{s}$
Y	5.85	m^3
	Attributes of the Component	
θ'_k	$min(a_k, \psi c_k)$	pcs
θ'_{ki}	$max(1, min(r_{ki}, a'_k))$	boxes
θ_k	$min(a_k, c_k)$	pcs
θ_{ki}	$max(min(\tau_{ki} c_k \psi, a_k), min(c_k, a_k))$	pcs
τ_{ki}	$[0.02, 1.05]$	$\frac{pcs}{s}$
a'_k	$[1, 2]$	boxes
a_k	$[1, 10]$	pcs
A'_k	*Obtained from the literature or through field observations*	m
A_k	*Obtained through field observations*	m
B'_k	*Obtained from the literature or through field observations*	m
B_k	*Obtained through field observations*	m
c_k	$[1, 5]$	pcs
h_k	$[0.01, 50]$	€
M_k	*Obtained through field observations*	kg
M'_k	*Obtained through field observations*	kg
p_k	*Obtained through field observations*	pcs
Vol_k	*Obtained through field observations*	m^3
Vol'_k	*Obtained through field observations*	m^3
r_{ki}	$\frac{\tau_{ki} c_k \psi}{p_k}$	boxes

These are the ranges used for the random generation of h_k, a_k, and a'_k. For each attribute, a random number based on the ranges of the attribute collected from company records is assigned. If there are multiple ranges in the attributes obtained from company records, the earliest one is accepted. These *features* are not part of the *training*, *validation*, and *testing sample* because we want to avoid that any undesired generated correlation affects the quality of the decision tree.

3.3 Modeling

3.3.1 Optimisation Model

The following Mixed Integer Linear Programming (MILP) model assigns a line feeding mode to each component. This model is a modified version of the one described in Sali and Sahin (2016) and considers one additional line feeding mode that was not originally presented. We describe the set and parameters, the decision variables, the objective function, and the constraints.

All the sets, parameters, and attributes of the components we introduce in the model are reported in Table 3.1. The depth and the length of the components (A_k and B_k) and of the *containers* (A'_k, B'_k, A^{sta}, and B^{sta}) are respectively considered for the depth and the length at the BoL (ξ and ζ). To give the reader a clear understanding, some of the attributes of the component and of the *containers* are depicted in Figure 3.2.

There is one primary binary decision variable x_{uk} that refers to the line feeding modes u of component k, with u = 1 for *stationary kit*, u = 2 for *sequencing*, u = 3 for *line stocking*, and u = 4 for *externally supplied sequencing*. Further, there are two auxiliary decision variables. N_n^{sta} is the number of *stationary kits* used in the delivery of components at station n, and m is the number of roundtrips performed by the tugger trains when they deliver the components.

$$\min \quad C_{prep} + C_{tr} + C_{BoL} + C_{st} + C_{hold} + C_{eq} \tag{3.1}$$

The objective function minimses the sum of all the costs for the line feeding procedures. We enhance the objective function by considering additional elements of cost, i.e. C_{hold} and C_{eq}. Furthermore, we consider an additional line feeding mode. The costs of the objective function are made up by the four line feeding modes and the six main cost elements described in Section 3.2.

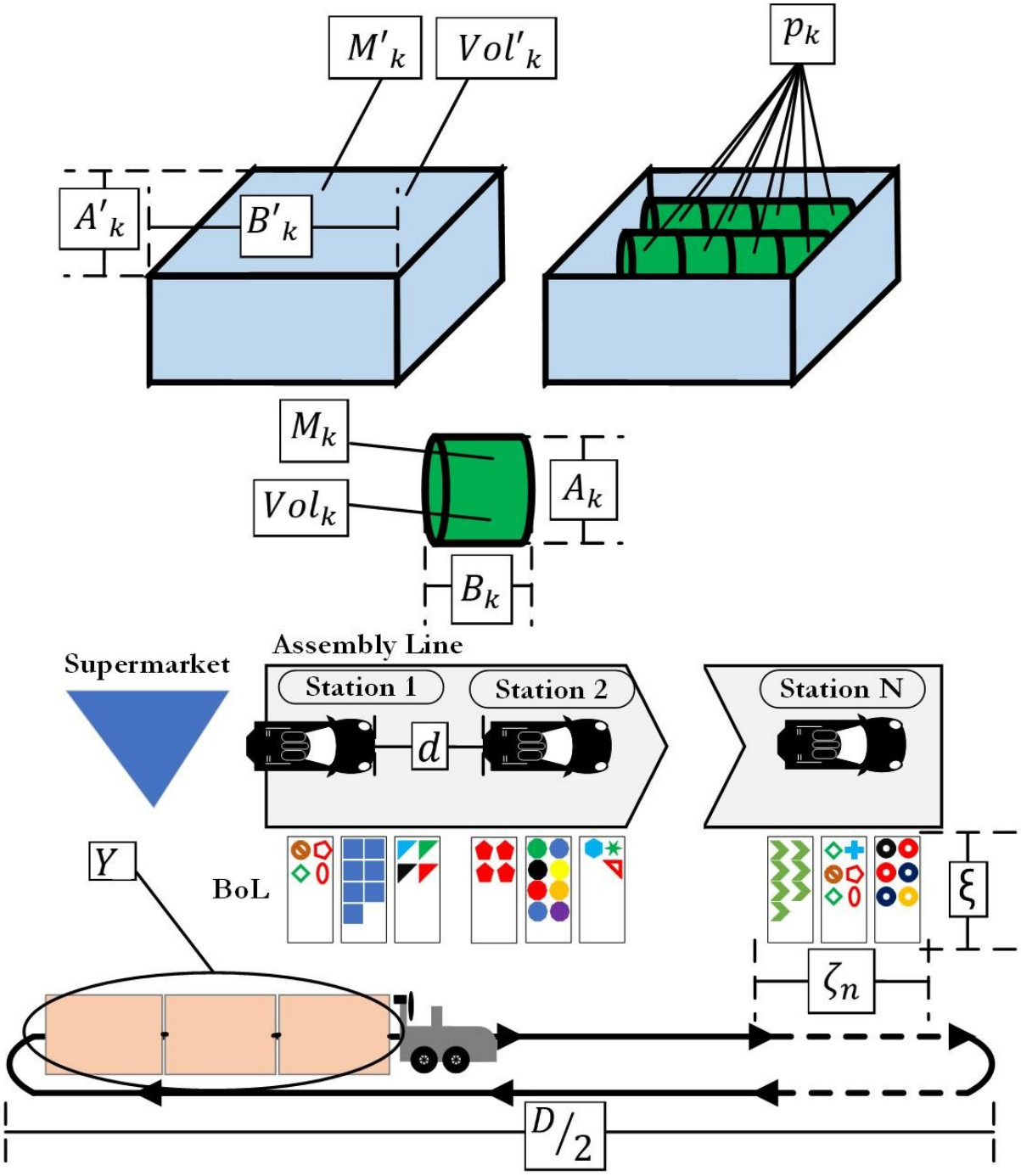


Figure 3.2: Dimensions of the assembly line and attributes of the components

Preparation Before Assembly

$$
\begin{aligned}
C_{prep} = & \underbrace{\frac{VC_0}{\psi 2 v_t} \sum_{k \in K} x_{3k} |S_k| B'_k}_{(3.2.1)} + \underbrace{\frac{VC_0}{\psi 2 v_0} \sum_{k \in K} (x_{1k} + x_{2k}) |S_k| B'_k}_{(3.2.2)} + \\
& VC_0 \Bigg\{ \sum_{k \in K} \sum_{i \in S_k} \underbrace{\left[x_{3k} \frac{\tau_{ki} c_k}{p_k \theta'_{ki}} t_{gl} \right]}_{(3.2.3)} + \underbrace{x_{1k} \frac{\tau_{ki} c_k}{\theta_{ki}} t_{pk} + x_{2k} \frac{\tau_{ki} c_k}{\theta_{ki}} t_{ps} \Bigg]}_{(3.2.4)} + \\
& \underbrace{\sum_{n \in N} \left[N_n^{sta} t_{lk} + N_n^{sta} t_{uk} \right]}_{(3.2.5)} + \\
& \underbrace{\sum_{k \in K} \sum_{i \in S_k} \left[x_{3k} \frac{\tau_{ki} c_k}{p_k \theta'_{ki}} t_{gul} + (x_{2k} + x_{4k}) \frac{\tau_{ki} c_k}{\theta'_k} t_{us} \right]}_{(3.2.6)} \Bigg\} + \underbrace{\psi \sum_{k \in K} x_{4k} C_{esc}}_{(3.2.7)}
\end{aligned}
\tag{3.2}
$$

The costs incurred in the preparation activities (C_{prep}) are considered in (3.2).
The total number of roundtrips ($\frac{V}{\psi}$) is multiplied by the time needed to collect the components from the supermarket. Since, for the *line stocking* mode, this operation is performed on a tugger train, the speed of the tugger train (v_t) is applied in (3.2.1). On the other hand, the speed of the operator (v_0) is used for the *kitting* and *sequencing* mode since the operator walks in the picking area, see (3.2.2). This formula is divided by 2 because we need to consider the U-shaped paths and the structure of the storage area.
Then, the costs for retrieving the components from the shelves are calculated in (3.2.3) and in (3.2.4). In the objective function and the constraints, we use the attributes of the components r_{ki}, θ'_{ki}, θ_{ki}, θ_k and θ'_k. For a detailed description of these attributes, we refer the reader to Sali and Sahin (2016). The difference between *line stocked components* in (3.2.3) and *kitted* and *sequenced parts* in (3.2.4) is the use of the number of components per *box* (p_k) and the average number of components k of variant i that can be picked up at once (θ_{ki}).
The *stationary kits* must be loaded onto and unloaded from the tugger train. The loading and unloading operations of the *stationary kits* in (3.2.5) are proportional to their numbers (N_n^{sta}). We use the loading time t_{lk} and the unloading time t_{uk}.
The *boxes* and *sequenced parts* must be unloaded from the tugger train. The unloading operations of *boxes* and *sequenced parts* in (3.2.6) are calculated in the same way as shown in (3.2.1) and in (3.2.2). We define the time for unloading operations of *line*

stocked components t_{gul} and the time for unloading operations of *sequenced parts* t_{us}. To obtain the duration of the operations, the time parameters (t_t) are used in the previous equations. To obtain the costs of these operations, the hourly labour cost (C_0) is multiplied by their duration.
For *externally supplied sequenced parts* components, there is a purchasing cost for having this activity performed by the supplier as shown in (3.2.7). *Externally supplied sequenced parts* do not need any preparation activity as they are delivered by the supplier ready for delivery to the BoL.

Transportation

$$C_{tr} = \frac{VD}{\psi v_t} m (C_0 + \frac{C_v}{T}) \tag{3.3}$$

The tugger train must perform a milk-run to the assembly line to replenish the racks at the stations. As shown in (3.3), the transportation cost (C_{tr}) is the same for all the four line feeding modes. It is calculated by multiplying the number of milk-runs (m) with the total number of roundtrips ($\frac{V}{\psi}$), the labour cost (C_0), the costs of the vehicle ($\frac{C_v}{T}$), and the distance travelled in one milk-run (D), and then divided by the speed of the tugger train (v_t).

Picking at the BoL

$$\begin{aligned} C_{BoL} = & \underbrace{VC_0 \sum_{k \in K} \sum_{i \in S_k} x_{3k} \tau_{ki} t_{il}}_{(3.4.1)} + \underbrace{\frac{V2C_0}{v_0} \sum_{k \in K} \sum_{i \in S_k} x_{3k} \frac{\tau_{ki} c_k}{\theta_k} (i-1) B'_k}_{(3.4.2)} + \\ & \underbrace{VC_0 \sum_{k \in K} \sum_{i \in S_k} \left[x_{3k} \frac{\tau_{ki} c_k}{\theta_k} t_{gsl} + x_{1k} \frac{\tau_{ki} c_k}{\theta_k} t_{pak} + (x_{2k} + x_{4k}) \frac{\tau_{ki} c_k}{\theta_k} t_{pas} \right]}_{(3.4.3)} \end{aligned} \tag{3.4}$$

The variants of components that are delivered under the *line stocking* mode must be identified as described in (3.4.1). Sali and Sahin (2016) provide a detailed description of how the time needed for identifying a variant i of a component k (t_{il}) is obtained. For *line stocked* components, the operator must travel from the starting point to the location of the *box* that is filled with a variant, see (3.4.2). The travelled distance is obtained from $(i-1)B'_k$. The number of trips made by the operator is considered in

$V\frac{\tau_{ki}c_k}{\theta_k}$, and the speed of the operator (v_0) is used for obtaining the traveling time. The number of parts picked in one go during the assembly is encoded in θ_k.
The operator needs to retrieve the components from the *containers* as shown in (3.4.3). The formula is similar to the one we used for retrieving the components in the supermarket as shown in (3.2.3) and in (3.2.4).
The labour cost (C_0) is used in the equations to obtain the costs for the working time.

Storage

$$
\begin{aligned}
C_{st} = & \underbrace{\sum_{k\in K}\sum_{i\in S_k} x_{3k}\lceil r_{ki}\rceil A'_k B'_k \frac{C_{m^2}}{L}}_{(3.5.1)} + \\
& \underbrace{\sum_{n\in N} \frac{\psi A^{sta} B^{sta} C_{m^2}}{2} N_n^{sta}}_{(3.5.2)} + \underbrace{2\psi C_{m^2} \sum_{k\in K}\sum_{i\in S_k} x_{2k}\tau_{ki}c_k A_k B_k}_{(3.5.3)} + \\
& \underbrace{\sum_{k\in K}(x_{1k}+x_{2k}+x_{3k})|S_k|A'_k B'_k \frac{C_{m^2}}{L}}_{(3.5.4)} + \underbrace{\sum_{k\in K}|S_k|Vol'_k C_{sh} x_{3k}}_{(3.5.5)} + \\
& \underbrace{\sum_{k\in K}|S_k|Vol'_k C_{sh}(x_{1k}+x_{2k}+x_{3k})}_{(3.5.6)}
\end{aligned}
\tag{3.5}
$$

The storage costs (C_{st}) are outlined in (3.5).
The dimensions of the *boxes* (A'_k and B'_k) and the number of *boxes* ($\lceil r_{ki}\rceil$) are considered to account for the space needed to store *line stocked* components at the BoL.
Kitted containers and *sequenced parts* must be stored in the preparation area close to the supermarket in (3.5.2) and (3.5.3). The dimensions of the *containers* (A^{sta}, B^{sta}) and (A_k, B_k) are multiplied by their numbers. The space needed at the supermarket is calculated in (3.5.4) for all the line feeding modes except for *externally supplied sequencing*.
In all these elements of cost, the occupied space is multiplied by the rental cost per square meter (C_m^2).
To account for the shelves where the *line stocked* components are stored at the BoL, their cost (C_{sh}) is multiplied by the size of the box in (3.5.5). For all the line feeding modes except *externally supplied sequencing*, the cost of the shelves at the supermarket

(C_{sh}) is multiplied by the volume of the box in (3.5.6).

Cost of Capital

$$C_{hold} = V \sum_{k \in K} \sum_{i \in S_k} \Big[\underbrace{x_{3k} \lceil \tau_{ki} \rceil c_k h_k j p_k}_{(3.6.1)} + \underbrace{(x_{1k} + x_{2k} + x_{4k}) \tau_{ki} c_k h_k j}_{(3.6.2)} \Big] \tag{3.6}$$

The cost of capital (C_{hold}) of the *containers* stored at the BoL are considered in (3.6). The quantity of material that is kept at the BoL is multiplied by its value h_k and by the current daily interest rate j.
While the whole *box* that contains the component k is considered for the *line stocked components* in (3.6.1), only the precise number of parts is calculated for the other line feeding modes in (3.6.2).

Equipment Costs

$$C_{eq} = 2(\sum_{n \in N} N_n^{sta}) C_{kit} + 2 \sum_{k \in K} x_{2k} C_{sq} \tag{3.7}$$

Some additional costs are incurred for the *containers*. Components that are *kitted* or *sequenced* require some additional equipment. The number of *kits* and *sequenced* components is multiplied by their costs (C_{kit} and C_{sq}) in (3.7). Since we assume that one *container* is used at the station and another *container* is used in the supermarket, we multiply the number of *containers* by 2. The constraints are arranged as *consistency constraints*, *takt time constraints*, *layout constraints*, *transportation capacity constraints*, and *variable definition*.

Consistency Constraints

$$\sum_{u \in U} x_{uk} = 1, \quad \forall k \in K \tag{3.8}$$

$$N_n^{sta} \geq x_{1k}, \quad \forall n \in N, \forall k \in S_n \tag{3.9}$$

Only one of the line feeding modes between *line stocking*, *stationary kitting*, *sequencing*, and *externally supplied sequencing* can be assigned to each component k. In (3.8), we ensure that only one of the binary decision variables from the line feeding modes of a component k is equal to 1 and the remaining ones are all equal to 0. If some components

are *kitted*, at least one *stationary kit* must be prepared and used as described in (3.9). The number of *stationary kits* depends on where they are located.

Takt Time Constraints

$$\begin{aligned}\frac{D}{v_t} + \frac{1}{2v_t}\sum_{k\in K} x_{3k}|S_k|B'_k + \psi\sum_{k\in K}\sum_{i\in S_k} x_{3k}\frac{\tau_{ki}c_k}{p_k\theta'_{ki}}t_{gl} + \\ \psi\sum_{k\in K}\sum_{i\in S_k} x_{3k}\frac{\tau_{ki}c_k}{p_k\theta'_{ki}}t_{gul} + \psi\sum_{n\in N} N_n^{sta}\,t_{lk} + \\ \psi\sum_{n\in N} N_n^{sta}\,t_{uk} + \psi\sum_{k\in K}\sum_{i\in S_k} x_{2k}\frac{\tau_{ki}c_k}{\theta'_k}t_{us} + \\ \psi\sum_{k\in K}\sum_{i\in S_k} x_{4k}\frac{\tau_{ki}c_k}{\theta'_k}t_{us} \le \frac{T\,\psi}{V}\end{aligned} \tag{3.10}$$

$$\frac{1}{2v_0}\sum_{k\in K} x_{1k}|S_k|B'_k + \psi\sum_{k\in K}\sum_{i\in S_k} x_{1k}\frac{\tau_{ki}c_k}{\theta_{ki}}t_{pk} \le \frac{T\,\psi}{V} \tag{3.11}$$

$$\frac{1}{2v_0}\sum_{k\in K} x_{2k}|S_k|B'_k + \psi\sum_{k\in K}\sum_{i\in S_k} x_{2k}\frac{\tau_{ki}c_k}{\theta_{ki}}t_{ps} \le \frac{T\,\psi}{V} \tag{3.12}$$

$$\begin{aligned}\sum_{k\in S_n}\sum_{i\in S_k} x_{3k}\tau_{ki}t_{il} + \frac{2}{v_0}\sum_{k\in S_n}\sum_{i\in S_k} x_{3k}\frac{\tau_{ki}c_k}{\theta_k}(i-1)B'_k + \\ \sum_{k\in S_n}\sum_{i\in S_k} x_{3k}\frac{\tau_{ki}c_k}{\theta_k}t_{gsl} + \sum_{k\in S_n}\sum_{i\in S_k} x_{1k}\frac{\tau_{ki}c_k}{\theta_k}t_{pak} + \\ \sum_{k\in S_n}\sum_{i\in S_k}(x_{2k}+x_{4k})\frac{\tau_{ki}c_k}{\theta_k}t_{pas} \le \frac{T}{V}, \quad \forall n \in N\end{aligned} \tag{3.13}$$

The time needed for transporting the components must be lower than the time interval between the deliveries to the stations in (3.10). The transportation time is the sum of:

- the traveling time for a milk-run;
- the traveling time needed in the preparation area to retrieve the *line stocked components*;
- the time it takes to load and unload the *line stocked components* onto and from the tow train;
- the loading and unloading time of the *stationary kits* onto the tugger train;
- the unloading time of the *sequenced parts* from the tugger train;

- the unloading time of the *externally supplied sequenced parts* from the tugger train.

Since it is assumed that, if necessary, more than one tugger train can deliver the components, the number of milk-runs m does not appear in the constraint. One *stationary kit* is prepared between two consecutive deliveries to the stations. Therefore, (3.11) sets the time the operator needs for preparing the *stationary kit* lower than or equal to the time interval between deliveries. This time includes the traveling time in the preparation area and the time needed for the picking activities in the preparation area. The *sequenced parts* are prepared between two consecutive deliveries to the stations. In the same way as described in (3.11), (3.12) sets the time the operator needs for preparing the *sequenced parts* lower than or equal to the time that elapses between two deliveries to the stations. It is not necessary to include such a constraint for *externally supplied sequenced parts* since they are provided to the assembly line by the supplier in the state that is desired for delivery. For each assembly station, (3.13) ensures that the time needed for retrieving the components at the BoL is lower than or equal to the takt time. This time is the sum of five elements:

- the time needed to identify a *line stocked* component;
- the traveling time at the BoL;
- the time needed to grasp a *line stocked* component;
- the picking time for a *stationary kit*;
- the picking time for *sequencing* and for *externally supplied sequencing* components.

Layout Constraints

$$x_{3k}|S_k|\frac{B'_k}{L} + N_n^{sta}B^{sta} + \sum_{k\in S_n}(x_{2k} + x_{4k})B_k + \leq \zeta_n, \quad \forall n \in N \tag{3.14}$$

$$x_{3k}\lceil r_{ki}\rceil A'_k \leq \xi, \quad \forall k \in K, \forall i \in I \tag{3.15}$$

$$x_{2k}\psi A_k \leq \xi, \quad \forall k \in K \tag{3.16}$$

$$N_n^{sta}A^{sta} \leq \xi, \quad \forall n \in N \tag{3.17}$$

$$x_{4k} = 1, \quad \forall k \in K : |S_k| \geq L_{seq} \tag{3.18}$$

$$x_{4k} = 1, \quad \forall k \in K : Vol'_k > \omega \wedge |S_k| > 1 \tag{3.19}$$

The available space in each workstation at the BoL must be big enough to hold all the *containers*. Therefore, the BoL must be long enough to hold the length of the *line stocked components*, *stationary kits*, *sequenced components*, and *externally supplied sequenced* components in (3.14). This is obtained by the sum of:

- the length of the *line stocked* components divided by the number of levels of the shelves;
- the length of the *stationary kit*;
- the length of the *sequenced* and *externally supplied sequenced* components.

The *box* of each *line stocked* component cannot have a depth (A'_k) multiplied by the number of *boxes* ($\lceil r_{ki}\rceil$) that is higher than the maximum depth available at the BoL (ξ), as described in (3.15). The depth of each *sequenced part* and of the *externally supplied sequenced parts* (A_k) multiplied by the delivery batch size (ψ) must be lower than the depth available at the BoL (ξ), otherwise they cannot be stored there (3.16). Following the procedure outlined in (3.16), the depth of the *stationary kit* must be lower than or equal to the depth at the BoL (3.17). This constraint has been newly introduced as it cannot be found in the original model presented by Sali and Sahin (2016), which is due to the fact that the original paper only considers *traveling kits*. If a component has more variants than the maximum limit L_{seq}, then this component is *externally supplied sequenced* in (3.18). This is due to the limited size of the preparation area where the *sequencing* activities occur. Since some components might have an enormous number of variants, the preparation activities for *sequenced* components cannot be performed.

Similarly, if a component's box has a higher volume than the limit ω and more than one variant, then it must be *externally supplied sequenced* in (3.19).

Transportation Capacity Constraints

$$x_{1k}A_k \leq A^{sta}, \quad \forall k \in K \tag{3.20}$$

$$x_{1k}B_k \leq B^{sta}, \quad \forall k \in K \tag{3.21}$$

$$x_{2k} \leq (|S_k| - 1), \quad \forall k \in K \tag{3.22}$$

$$x_{3k} = 1, \quad \forall k \in K : Vol'_k \leq 0.0208 \tag{3.23}$$

$$x_{1k}c_kM_k \leq M^{sta}, \quad \forall k \in K \tag{3.24}$$

$$x_{1k}c_kVol_k \leq Vol^{sta}, \quad \forall k \in K \tag{3.25}$$

$$\sum_{k \in S_n} x_{1k}c_kM_k \leq N_n^{sta}M^{sta}, \quad \forall n \in N \tag{3.26}$$

$$\sum_{k \in S_n} x_{1k}c_kVol_k \leq N_n^{sta}Vol^{sta}, \quad \forall n \in N \tag{3.27}$$

$$\sum_{k \in K}\sum_{i \in S_k} x_{2k}c_kVol_k\tau_{ki} + \sum_{k \in K}\sum_{i \in S_k} x_{4k}c_kVol_k\tau_{ki} + \sum_{n \in N} N_n^{sta}Vol^{sta} + \sum_{k \in K}\sum_{i \in S_k} x_{3k}\frac{\tau_{ki}c_k}{p_k}Vol'_k \leq \frac{m}{\psi}Y \tag{3.28}$$

If a component is bigger than the *kit container*, then it cannot be delivered as *kitting*, as in (3.20) and (3.21). *Sequencing* is a line feeding mode used for components that have multiple variants. If a component has only one variant, then it cannot be *sequenced* as described in (3.22). The National Institute for Occupational Safety and Health (NIOSH) guidelines and International Organization for Standardization (ISO) standards suggest an equation that can be used for ergonomically improving the repeated lifting and carrying operations of *boxes* (ISO 11288-1:2003 2003; Waters, Putz-Anderson, and Garg 1994). Through this equation, we can assert that the appropriate size when it is necessary to transport a *box* manually is 0.0208 m^3. In (3.23), we enforce that *line stocking* is assigned to all those components that have a volume lower than or equal to the recommended optimal size. The weight of each component that is *kitted*, which consists of the weight of a single part (M_k) multiplied by its bill of material (BOM) coefficient (c_k), must not exceed the maximum weight of the *kitted containers* (M^{sta}). (3.24) refers to the *stationary kit*. The volume of each component that is *kitted*, which consists of the volume of a single part (Vol_k)

multiplied by its BOM coefficient (c_k), must not exceed the maximum volume of the *kitting containers* Vol^{sta}. (3.25) refers to the *stationary kit*. The total weight (3.26) and volume (3.27) of components that are *kitted* must be lower than or equal to the maximum available weight and volume of the *stationary kit*. To obtain the total weight and volume of the *kit containers*, their number N_n^{sta} is multiplied by the maximum weight and volume of a single *kit*. The volume of the *containers* to be delivered must be lower than or equal to the available volume of the tugger trains in (3.28). The volume of a single tugger train (Y) is multiplied by the number of milk runs (m) to obtain the total available volume. The transported volume is the sum of four elements:

- the sum of the volumes of all variants of the *sequenced* component;
- the sum of the volumes of all variants of the *externally supplied sequenced* component;
- the volume of the *stationary kits*;
- the sum of the volumes of the *line stocked* components.

Variable Definition

$$\begin{aligned} x_{uk} \in \{0,1\},\ \forall u \in U,\ \forall k \in K \\ N_n^{sta} \in \mathbb{N}, m \in \mathbb{N} \end{aligned} \tag{3.29}$$

The decision variables are defined in (3.29). While x_{uk} is a binary decision variable, the auxiliary decision variables are set as integer.

3.3.2 Supervised Machine Learning Process

Three sets of data are needed in the machine learning approach: the *training sample*, the *validation sample*, and the *testing sample*. Since this is a Supervised Machine Learning algorithm, the *training sample* is used for training the machine learning algorithm. This *sample* is made up of P instances which are solved with the optimisation model. After the instances are solved, each problem, along with its solution, is used for the *training* phase of the machine learning approach. The four line feeding modes chosen by the optimisation approach represent the four *classes* into which the components are *classified*. The classification accuracy is calculated as the percentage of *classes* correctly chosen by the machine learning algorithm.

To develop a decision tree, a gini score is calculated. It was first described in Gini (1912), this coefficient represents the degree of equality among the data, where 1 expresses total inequality and 0 total equality.

$$G_f = 1 - \sum_{c=1}^{C} \psi_{fc}^2 \tag{3.30}$$

Given a node f of a decision tree, the gini impurity score can be obtained from Equation (3.30). ψ_{fc} encodes the number of instances (components) of *class* (line feeding mode) c in the sample at node f divided by the total number of instances of the sample, i.e. the total number of components of the sample at node f.
To *train* the decision tree, we implement the Classification And Regression Tree (CART) (Breiman 1984) because this algorithm outperforms the Iterative Dichotomiser 3 (ID3). The CART is also comparable in performance to the C4.5 algorithm (Gokgoz and Subasi 2015). Instead of implementing a gradient boosting algorithm that minimses the costs, we choose the CART algorithm that maximizes the classification accuracy and we later use a repair approach to ensure the feasibility of the instances and to reduce the cost deviation from the optimal solution of the MILP.
At each node, rules that split the *sample* based on the values of the *features* are generated. The rules are evaluated and the one that minimse the CART-objective function is selected.

$$\min \frac{m_l}{m} G_l + \frac{m_r}{m} G_r \tag{3.31}$$

In the CART-objective function (3.31), m, m_l, and m_r are the number of instances of the sample at a node at the sequent left node l and at the sequent right node r. G_l and G_r are the gini impurity scores of the sequent left l and the right node r. In other words, the rule that best splits the *classes* (line feeding modes) is selected.
After the rule has been generated, this leads to a splitting of the first node into two nodes and of the *sample* into two subsamples. This process is iterated for each node until the end of the tree is reached, i.e. until the maximum depth of the tree is reached or the minimum *sample* for generating a node or leaf node is obtained. As a final step, the newly generated tree is tested to increase the classification accuracy by pruning the tree, i.e. eliminating one or several final nodes.
In a decision tree, each node represents a decision that must be made based on the *features* of the component. After the system has moved along the nodes of the decision tree, a leaf node is reached. This node represents the line feeding mode that must be

assigned. For each node of the decision tree, there is a gini impurity score (Gini 1912). As described in Bischl et al. (2016), *hyperparameters* are settings of a machine learning process that are selected before the learning process begins. They can be tuned in such a way that they develop a decision tree with the highest classification accuracy. James et al. (2017) describe the use of the *validation sample* when it comes to tuning such *hyperparameters*. In our case, the *validation sample* is made up of P' instances that are solved with the optimisation model. To reach the highest classification accuracy on the *validation sample*, we iteratively vary three *hyperparameters*: the maximum number of levels of the decision tree, the minimum size of the *sample* to generate a node, and the minimum size of the *sample* to generate a leaf node. This allows the generation of the best decision tree for *classifying* the components.

For the *testing sample*, another set of P'' instances is solved with the optimisation model. After the machine learning algorithm is trained with the *training sample* and the *hyperparameters* are tuned with the *validation sample*, a test is made on the *testing sample* with the same settings as before. The test determines the classification accuracy of the model. As opposed to the *training sample*, only the *features* and not the *classes* of the *validation* and *testing samples* are given to the model. The model will then *classify* the components and return their *classes*.

To ensure that the results are meaningful, we apply the Leave-p Out Cross-validation (Celisse and Robin 2008). In this approach, iterative experiments are performed. During the experiments, the *samples* are permuted and used for the *training*, *validation*, and *testing phases*. This becomes a Leave-2 Out Cross-validation Approach since two *samples* are used for the *training* and *validation* phases, whereas only one *sample* is used for the *testing phase*. The out-of-sample test is performed because we wish to ensure that the model will perform well even with new data. On the other hand, we always ensure that the outliers are part of the *testing sample* and not included in the *training sample* or the *validation sample*.

To ensure that the model will perform well even in abnormal situations, we also generate 5% of instances where the four *classes* are abnormally present, i.e. one *class* occurs at least 90% of the time in each instance. We call these instances *outlier instances*. We implement Dixon's Q-test, as originally described in Dixon (1950). Although the original approach is preferable to other derived techniques, it is difficult to implement because of the limited sample volume. This is why we rely on an approach that has been tested on larger samples (Verma and Ruiz 2006). We select the *outlier instances* with a critical Q-value of 99.5%.

3.3.3 Repair Approach

As a result of the machine learning approach, each component is *classified* with a line feeding mode that was selected from the decision tree. However, sometimes the classification leads to an infeasible solution of an instance p. One component can affect the line feeding selection of the others, and this can lead to a violation of the volume or the space capacity. Therefore, even one single wrongly *classified* component can lead to an infeasible solution. For example, one component's *box* might be too large to be *line stocked* or have too many components per *box* to be *kitted*. If an incorrect line feeding mode is assigned to such a component, some of the constraints of the optimisation model might be violated. On the other hand, a component that is assigned to an incorrect line feeding mode might lead to a feasible solution but negatively affect the overall costs. If only one component is *kitted*, then a feasible solution with higher costs is a possibility because of the transportation costs of one nearly empty *kit*. Eventually, a component that is assigned to an incorrect line feeding mode might make an instance difficult to compare to others. If an instance is infeasible, it might have an objective value that is lower than the optimum. This is because one or more constraints are violated. Therefore, it is misleading to consider the objective value of an instance that is not repaired, i.e. before the repair approach is applied. There are multiple benefits of implementing a repair approach, ensuring the feasibility of the instances, ensuring a low cost deviation, and ensuring that the instances are comparable. The repair approach discussed in this section is applied for all instances regardless of their status. Our repair approach ensures the feasibility of the solution for each instance. In order to ensure the feasibility of an instance, we assign a line feeding mode to one component that violates a constraint by using the MILP explained in Section 3.3.1. For constraints that consider multiple line feeding modes (3.10 - 3.14), the component that generates the highest value in the constraint is selected. This allows the identification of those components that contribute towards violating a constraint.

After, it is possible that the cost deviation from the optimal solution of a feasible instance is high. To reduce such cost deviation, we assign a line feeding mode to the most expensive components by using the MILP previously explained. Since in the objective function there are some fixed costs for some line feeding modes, we distribute these fixed costs among all the components that are assigned to that *container*. Although this approach is not common, a standard heuristic that employs techniques such as the Nearest Neighbor Search is not a viable option for solving this problem because of the high level of constraints and the high number of permutations of the

line feeding mode. This can also be justified by the fact that in the literature the ALFP is always solved with optimisation models. Figure 3.3 provides the flowchart of the repair approach.

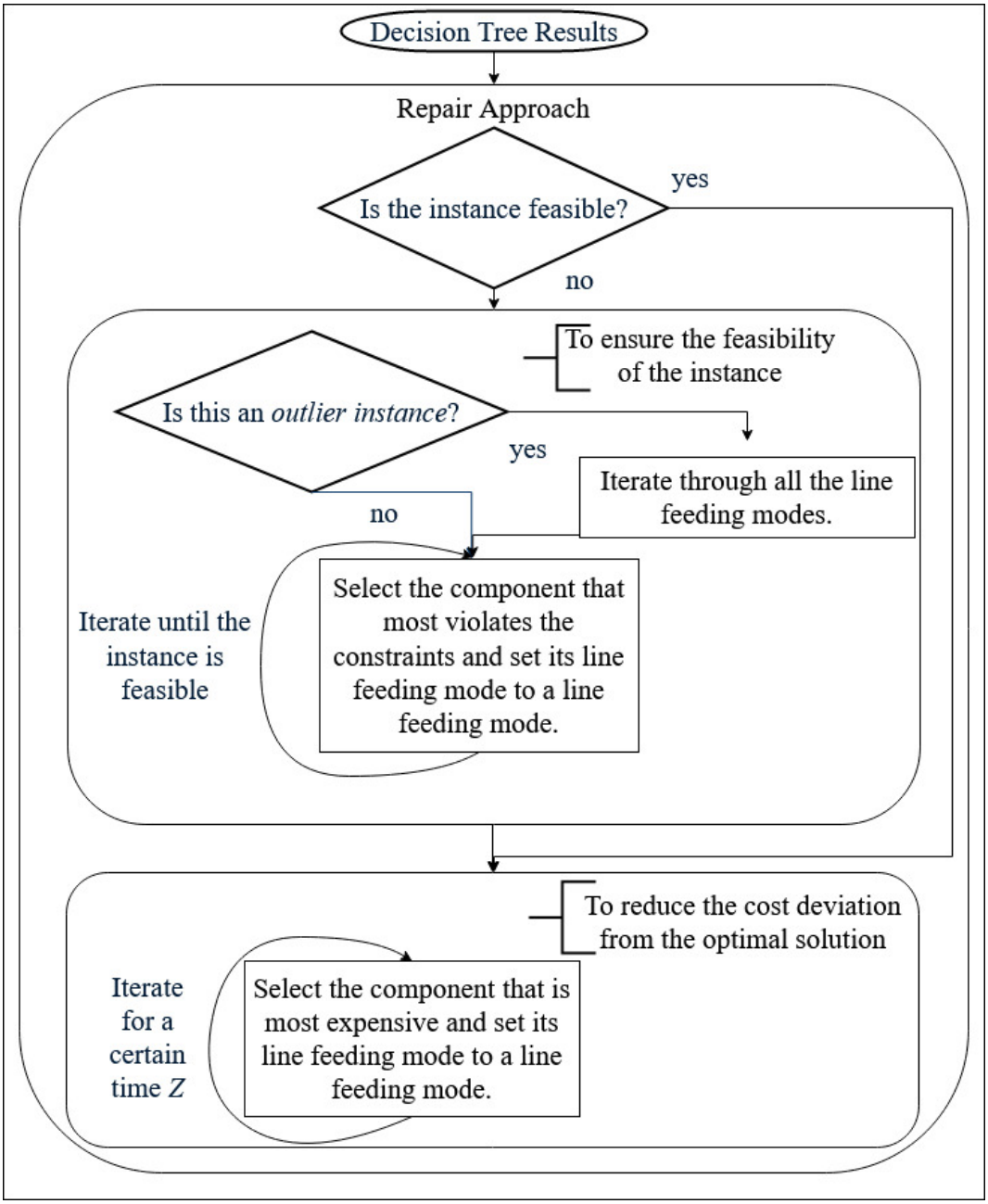


Figure 3.3: Flowchart of the repair approach

3.4 Numerical Study

3.4.1 Key Performance Indicators

We use four KPIs for evaluation: *the classification accuracy* represents the percentage of components that are assigned to the correct line feeding mode. *The confusion matrix* analyses the classification accuracy for each *class* relative to the other *classes*. For each line feeding mode u_r, we calculate the percentage of components *classified* by the decision tree in each line feeding mode u_c. The value at row u_r and at column u_c represents the percentage of components originally assigned to the line feeding mode r by the optimisation model and by the classification tree to the line feeding mode c. This allows the reader to understand what percentage of each line feeding mode is *classified* in the other line feeding modes. *The number of infeasible instances before the repair approach* represents the number of instances that become infeasible when the components are assigned a line feeding mode from the decision tree. Although each component has a certain line feeding mode, it is possible that all the K components together lead to a problem p that is infeasible in its complexity. *The total cost deviation after the repair approach* refers to the difference between the optimal value of the cost function and the solution obtained from the repair approach, which ensures that each instance p is feasible and that the cost deviation is low. This is the cost that can be calculated from the objective function (3.1). As said before, it is not possible to consider the cost deviation before an instance is repaired. This is because the instance might be infeasible, i.e. some constraints might be violated. When this occurs, the objective value might be lower than the optimal solution. This value might not bring a valid mathematical meaning. Thus, it is not possible to consider the total costs of an instance when only the *classification* of the decision tree is applied.

3.4.2 Data Generation

Our approach is tested with data that was obtained from the literature and collected through field observation. We took the parameters in Table 3.1 from Sali and Sahin (2016), Limére, Van Landeghem, and Goetschalckx (2015), and Caputo, Pelagagge, and Salini (2018) in order to develop a general decision tree. These data refer to operations and tools that are universally performed. Table 3.2 provides the values, the lists, or the ranges.

The *training sample*, the *validation sample*, and the *testing sample* are generated by using an approach that is based on the Resample and Replacement principle (Efron and Tibshirani 1993). This allows the generation of scenarios even when the observed data's statistical distribution cannot be estimated, is unknown, or lacks accuracy. Given a population G of dimension α and size γ, a random sample $\mu_1, \mu_2, \ldots, \mu_n$ of size n is obtained. This is achieved by a selection of n random integer numbers between 1 and γ, each with a probability $\frac{1}{\gamma}$. Since this is an approach with replacement, the integer numbers can be repeated. This will lead to $\mu_1 = G_1, \mu_2 = G_2, \ldots, \mu_n = G_n$. Each element μ_n of the new sample has the same dimension α of the population G.
Firstly, the *features* in Table 3.1 that are written in *italics* are collected from documents and records provided by four companies. The dimensions of the *boxes* underlined in Table 3.1 are obtained from the literature. Hemminki, Leipala, and Nevalainen (1998) describe an algorithm that optimally places *boxes* in a pallet where the sizes of the *boxes* are not previously known. This work presents a list of 260 *boxes* with their sizes. These *box* sizes are considered for the *testing*, *validation*, and *training sample* for every set of data that does not already include them. We use the previously described sampling technique to generate the instances for these *features*.
Subsequently, the remaining *features* are generated by assigning a random value to the ranges described in Table 3.2. Since we want the *features* not provided in any set of data (h_k , a_k, and a'_k) to have realistic values, we generate random values based on one or multiple *features* that were previously created with the Resample and Replacement approach. The ranges are described in Table 3.3. These and all other randomly generated *features* are not part of the *training*, *validation* and *testing sample* of the decision tree because we want to avoid that high levels of correlation — that might be an undesired byproduct of this process — lead to higher levels of accuracy and unrealistic results. This process generates instances that are similar to the data collected from the companies.
For the *outlier instances*, the generation process is the same as the one already presented, and we always append these instances only to the *testing sample*.

Table 3.3: Ranges used for the random generation of attributes of the components

Attributes obtained from company records				Randomly generated attributes	
p_k	$(1000pcs; +\infty)$			h_k	$[0.01€; 1€]$
	$[1pcs; 1000pcs]$				$[1€; 50€]$
M'_k	$(8kg; +\infty)$			a'_k	$\{1\text{pc}\}$
	$(0kg; 8kg]$				$[1\text{pc}; 2\text{pcs}]$
Vol_k	$(0.01m^3; +\infty)$	M_k	$(10kg; +\infty)$	a_k	$\{1\text{pc}\}$
	$(0.001m^3; 0.01m^3]$		$(5kg; 10kg]$		$[1\text{pc}; 2\text{pcs}]$
	$(0.0005m^3; 0.001m^3]$		$(1kg; 5kg]$		$[1\text{pc}; 3\text{pcs}]$
	$(0; 0.0005m^3]$		$(0kg; 1kg]$		$[1\text{pc}; 10\text{pcs}]$

3.4.3 Results and Discussion

We used a computer with a 64-bit Operative system, 22 GB RAM, and a CPU with 2.10 GHz to perform the computational experiments. The data generator was implemented with a combination of R Studio and Excel, while the optimisation model and the repair approach were realised with Xpress-IVE version 8.5. The machine learning algorithm was coded with Scikit-learn, which is a toolbox of Python 3.6.5 64 bit, and was executed with Spyder 3.3.1.

For the computation times, we provide the *validation time*, which represents the time needed for tuning the *hyperparameters*, and the *learning time*, which is the time needed for the *training* and *testing* phases. The *validation time* is 14 minutes and the *learning time* is 15 seconds. For the repair approach, we set the time Z for improving each instance with 100, 150 and 200 components K, which equals to 3, 4, and 5 seconds.

The sets of data for this numerical case were synthetically generated by applying the data generation approach described in Section 4.4.2 to data of four real-world industrial scenarios. The first case comes from an automotive manufacturer of sports cars, the second one comes from an OEM that produces components for the automotive environment, the third from a company that produces electrical motors, and the fourth from a manufacturer of components for nautical applications. We varied the number of components K and the number of stations N.

As discussed in Sebban et al. (2000), the out-of-sample accuracy of decision trees increases with the *sample* size. With the same analysis, we show in Figure 3.4, Figure 3.5, and Figure 3.6 how the KPIs are affected by the *sample* size.

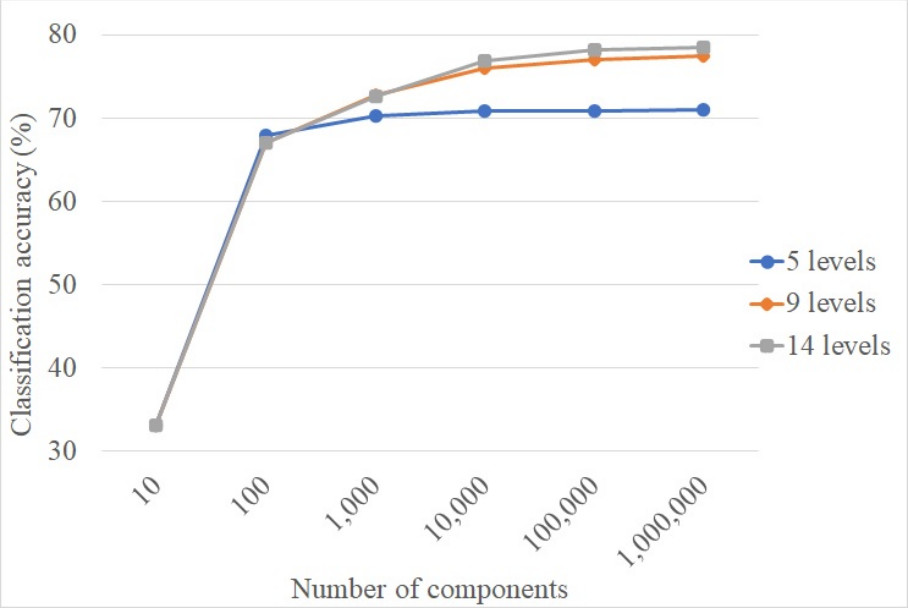


Figure 3.4: Classification accuracy by *sample* size

In Figure 3.4, the classification accuracy increases with the sample size. With a sample size of 100 000 components, the classification accuracy reaches the asymptotic limit. If the sample size increases further, the classification accuracy increases but only slightly. Although these charts are, in the literature, designed to select the sample size for the decision tree with the logarithmic scale, we rely on a linear scale in Figure 3.5 and Figure 3.6 so that we can refer to the instances solved. The classification accuracy and the average cost deviation do not improve with a *sample* size larger than 100 instances, but rather remain constant. In other words, the classification accuracy and the average cost deviation with 200 instances don't show a high improvement from the ones of 100 instances. On the other hand, the classification accuracy suffers slightly when we only consider 25 instances. The number of insfeasible instances increases with the *sample* size, but this is due to the higher number of instances of the *testing sample*. Therefore, we identify a *sample* size of 100 instances as the one that preserves the classification accuracy and avoids excessive computation times. Since we perform a Leave-2 Out Cross Validation approach, each set is permuted and used for the *training*, *validation*, and *testing* phases. One might argue that a *sample* P of only 100 instances is quite small for a supervised machine learning approach. On the other hand, each instance p consists of multiple components, and the *training*, *validation*, and *testing samples* have a multitude of sets of data. Consequently, each *sample* has 540 000 rows, each representing a component.

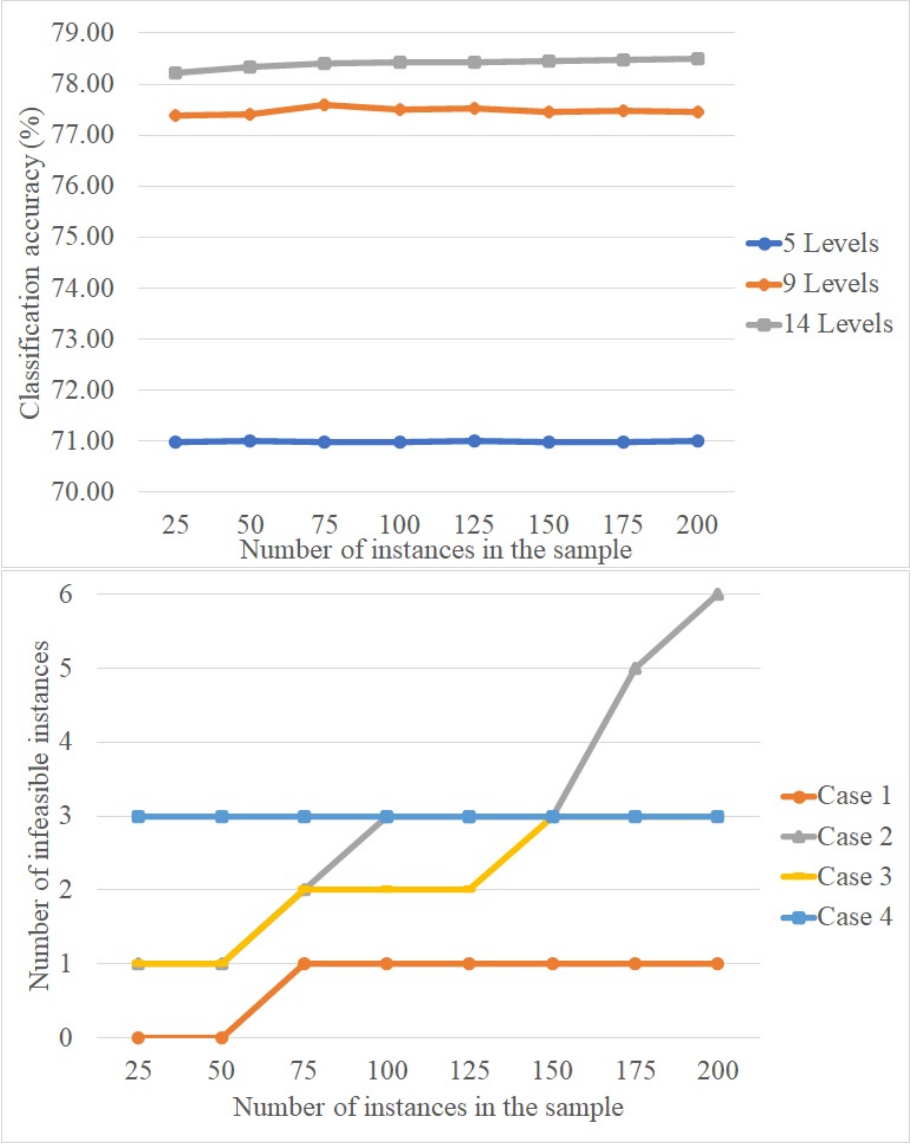


Figure 3.5: Classification accuracy and number of infeasible instances before the repair approach by *sample* size

Figure 3.7 presents the box plots of the occurrences with the percentages for the four *classes*, i.e. the four line feeding modes, in the *testing sample*. The box plots reveal that, for all four line feeding modes, the percentage has a median that is above 20% in at least one case. The percentage of occurrence of the *sequencing* and *stationary kit* are complementary since the components that are assigned to these line feeding modes are similar, i.e. they require preparation activities and are delivered without the use of the original *box*.

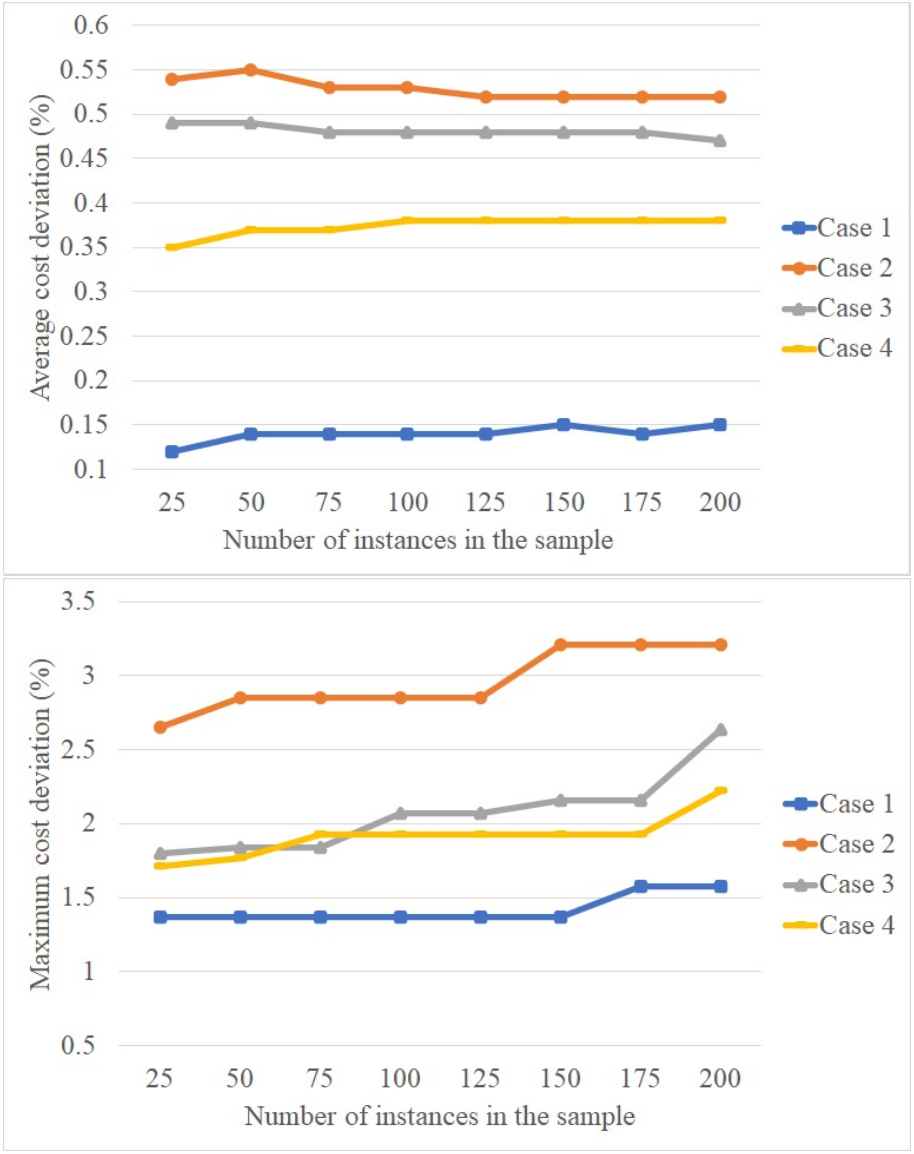


Figure 3.6: Average cost deviation and maximum cost deviation after the repair approach by *sample* size

For the *hyperparameters* of the decision tree, the minimum size of the *sample* to generate a node is 50 components, the minimum size of the *sample* to generate a leaf node is 15 components, and the maximum number of levels in the decision tree is 14. The number of levels of the decision tree refers to the maximum number of sequential nodes starting from the root node. With these *hyperparameters*, the classification accuracy is 78.49%.

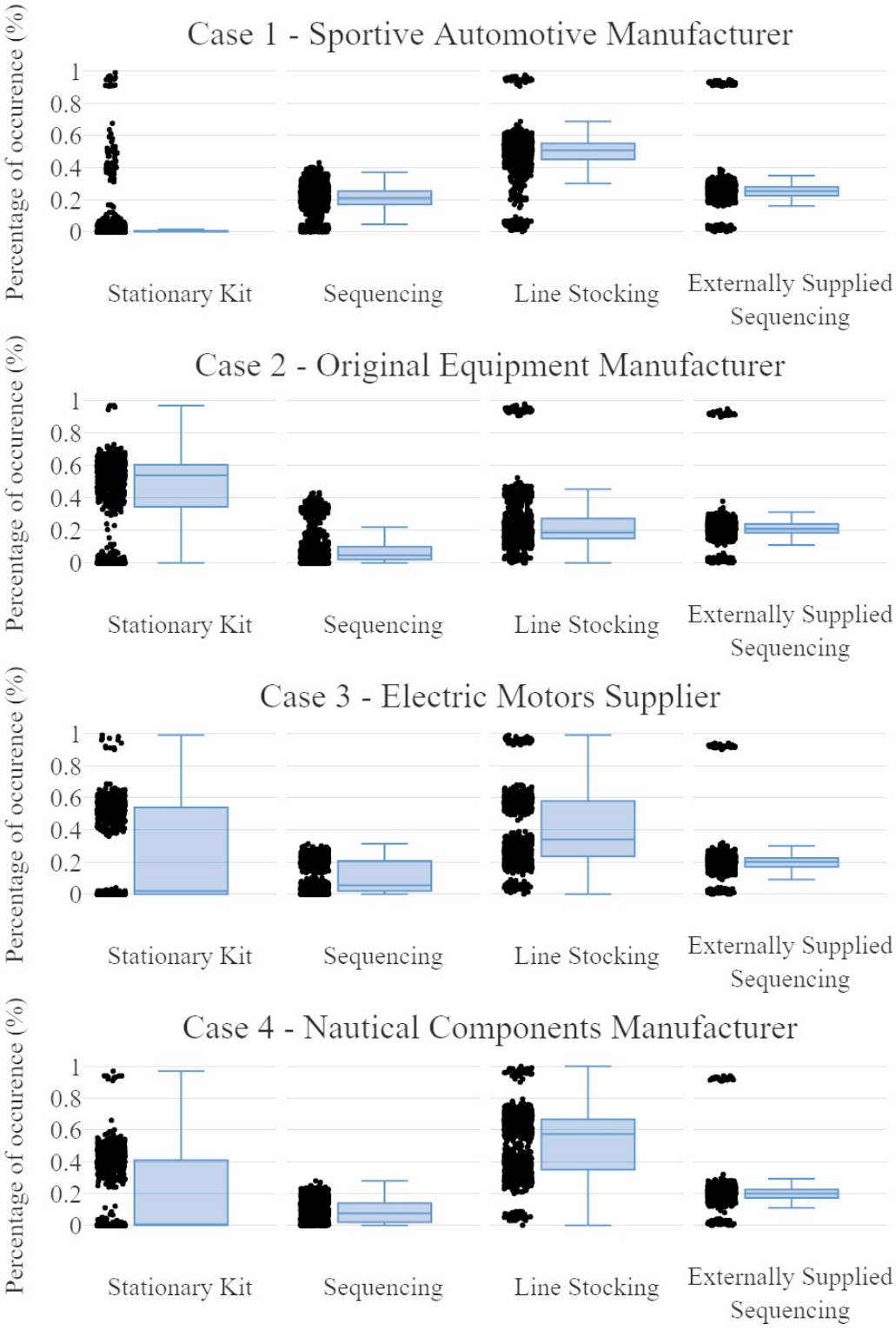


Figure 3.7: Occurrences of the *classes*.

Figure 3.8 shows how the classification accuracy changes with the maximum levels

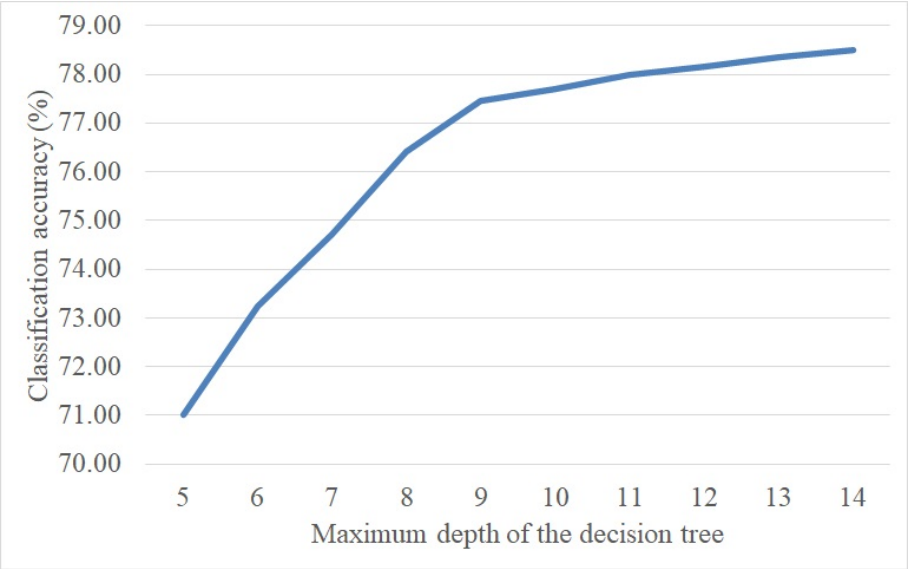


Figure 3.8: Classification accuracy by maximum depth of the decision tree

of the decision tree. The results show that even a tree with a low depth can have a classification accuracy of more than 70%. Table 3.4 presents the results of the decision tree and the repair approach by using the number of infeasible instances before the repair approach and the cost deviation from the optimal solution as previously described. Deviating from Figure 3.5 and Figure 3.6, we set the *sample* size P to 100 instances and provide the results when we variate the number of components K and of the stations N for all four cases. The results reveal that there is a minimum number of infeasible instances before the repair approach and that the cost deviation is low in all four cases. The number of components K and the number of stations N do not affect the quality of the results. There is no clear pattern that shows in which environment there is a minimum cost deviation and number of infeasible instances. In all four cases, the results indicate that we can predict the line feeding modes with a high degree of accuracy and that the repair approach is effective.

The confusion matrix in Table 3.5 shows how the *classes* obtained from the optimisation model are *classified* by the decision tree. The main diagonal of this square matrix represents the correct *classes*. The confusion matrix shows that the accuracy for the *kitted components* is lower than the ones for the other line feeding modes. Indeed, the box plots in Figure 3.7 show that, in one case, there is a fairly low occurrence of *kitted components*. Therefore, the occurrences of all four *classes* affect their *classification accuracies*. Although our approach performs quite well for all the remaining three line feeding modes and across all the four cases, this is a limitation of our approach.

Table 3.4: Number of infeasible instances before the repair approach, average cost deviation, and maximum cost deviation for a *sample* P of 100 instances

K Components	100			150			200		
N Stations	10	15	20	10	15	20	10	15	20
Number of infeasible instances before the repair approach									
Case 1	0	0	0	0	0	0	0	0	1
Case 2	0	0	0	0	1	0	1	0	1
Case 3	0	0	0	1	0	0	1	0	0
Case 4	0	0	0	0	0	0	1	1	1
Average cost deviation after the repair approach (%)									
Case 1	0.12	0.13	0.14	0.15	0.16	0.13	0.16	0.13	0.14
Case 2	0.01	0.02	0.04	0.72	1.22	1.53	0.11	0.25	0.88
Case 3	0.81	0.76	0.68	0.57	0.46	0.18	0.43	0.23	0.18
Case 4	0.70	0.63	0.57	0.35	0.28	0.17	0.27	0.17	0.24
Maximum cost deviation after the repair approach (%)									
Case 1	0.82	0.80	0.53	1.12	0.95	1.06	1.37	0.65	0.90
Case 2	0.19	0.22	0.26	1.41	2.44	2.85	0.82	1.36	2.26
Case 3	1.80	1.84	1.44	2.07	1.80	1.16	1.60	1.64	1.15
Case 4	1.93	1.20	1.28	1.71	1.93	1.11	1.49	1.10	1.77

3.4.4 Comparative Analysis of the Machine Learning Algorithms

Many machine learning approaches that are available can be implemented for classification problems. We compare these machine learning approaches in order to show that the decision tree is the machine learning approach that yields good classification accuracy. Although not all of them can provide an explanation for the classification, all of them can be used for classification problems. In this section, we aim at comparing these machine learning algorithms to show which one leads to the highest classification accuracy. The machine learning algorithms were all coded with Scikit-learn, which is a toolbox of Python 3.6.5 64 bit, and was executed with Spyder 3.3.1.

Eight machine learning approaches, that can all be used for *classification* problems, are compared:

- Decision tree is implemented as the approach explained in Section 3.3.2. The algorithm applied CART described in Breiman (1984).
- Logistic regression is a technique that finds an application in machine learning. Although the name suggests that the approach is used for regression problems,

Table 3.5: Confusion matrix reporting the classification accuracy (%) by class for a *sample* P of 100 instances

Line Feeding Modes	$u = 1$	$u = 2$	$u = 3$	$u = 4$
$u = 1$	48.09	15.11	36.80	0.00
$u = 2$	20.11	57.87	22.03	0.00
$u = 3$	6.81	4.60	88.58	0.00
$u = 4$	0.00	0.00	0.00	100.00

it is frequently implemented in two-class classification problems. For a multiclass problem, the method applied is the multinomial logistic regression. This technique provides a *classification* of the data without a graphical representation. To apply this classification technique, a different formulation of the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm (Fletcher 2008) is applied. Logistic regression is based on a Sigmoid function that returns values between 0 and 1 in (3.32).

$$f(x) = \frac{1}{1 + e^{-(x)}} \tag{3.32}$$

In this equation, we use a linear function $\alpha_0 + \alpha_1 x$ as in (3.33) where α_n are the coefficient known as predicted weights or estimators.

$$h(x) = \frac{1}{1 + e^{-(\alpha_0 + \alpha_1 x_1)}} \tag{3.33}$$

Based on the value returned by the function and the threshold, the elements are *classified*. The log-likelihood function (3.34) is maximized to calculate the estimators α_n through the limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm which is a variation of the homonymous algorithm (Fletcher 2008).

$$h(x) = \frac{1}{m} \sum_i [y \; log(h(x))] + (1 - y) log[1 - h(x))] \tag{3.34}$$

This algorithm relies on a function's gradient to maximize it. Differently from the original version, the L-BFGS stores a limited number of vectors that represent the Hessian matrixes.

- Naïve Bayes is a machine learning technique that is based on Bayes theory with a Naïve *classifier* which minimises the probability of a *classification* error. This technique assumes that the *features* have a Gaussian likelihood, i.e. the value of a *feature* does not depend on the value of any other *feature* (Chan, Golub, and Leveque 1982). In other words, any *feature* contributes independently to the assignment of an element to a *class*. There is a group of algorithms that can be implemented for Naïve *classification*. The algorithm used for this machine learning technique is detailed in Zhang (2004) and Zhang and Su (2008). Naïve Bayesian is an approach that assumes that the likelihood of the *features* is Gaussian. It assumes that the *features* have an independent Gaussian distribution. The probability that the element belongs to *class* u given a feature x_i in (3.35).

$$P(x_i \mid u) = \frac{1}{\sqrt{2\pi\sigma_u^2}} e^{(-\frac{(x_i-\mu_u)^2}{2\sigma_u^2})} \tag{3.35}$$

  The parameters σ_y and μ_y depend on the *class* y and are calculated using the maximum likelihood estimation.

- Nearest Centroid is a *classification* approach that is based on the idea that a *class* is represented by its centroid, i.e. its mean. Unlike other machine learning approaches described here, this technique can only be applied to *classification* problems. In the *training phase*, the centroids are calculated based on the euclidean distance between the points of the classes. Like in a Voronoi diagram, a centroid is the average position of the elements of a class. During the *training phase*, the centroid for each *class* u is calculated with the euclidean distance between the elements of a *class* in (3.36).

$$\mu_u = \frac{q}{|C_u|} \sum_l x_l \tag{3.36}$$

  During the *testing phase*, the elements of the *testing sample* are assigned to the *class* u with the nearest centroid in (3.37).

$$\min|\mu_u - x_l| \tag{3.37}$$

  This model was described and applied in medical applications (Tibshirani, Narasimhan, and Chu 2002). Rocchio algorithm is applicable for the Nearest Centroid approach.

- Random forest is a machine learning technique that relies on multiple decision trees for the *classification* problem. This technique consists in developing multiple decision trees each of them assigning a *class* to the data. The classification is then based on the *class* that is assigned with the highest percentage, i.e. the *class* that is assigned by the highest number of decision trees in the forest. Although each decision tree that is part of the forest can have a graphical representation, the approach does not provide a single decision tree that can be implemented in all cases. Furthermore, it does not provide a single and universal graph. This approach was originally described in Breiman (2001). Random forest is an approach that aims at developing multiple decision trees (Breiman 2001). The algorithm is similar to the one of the CART. However, a number of elements is drawn from the *training sample* using resample and replacement to train each random decision tree. Furthermore, another difference from the original algorithm is that the *classification* of the random decision trees is averaged by their probabilistic prediction.

- Support Vector Machine (SVM) consists of a multi-class version of the original two-class algorithm (Knerr, Personnaz, and Dreyfus 1990). SVM employs Support Vector Classification (SVC) for the classification problem (Cortes and Vapnik 1995). During the *training phase*, the technique identifies vectors in the hyperplane that divide the elements of the *classes*. In this case, the algorithm builds a model that performs one-against-one classification for each pair of *classes*. During the *testing phase*, the vectors of the hyperplane are used to *classify* the components. SVM involves a different version of the original algorithm. Given U *classes*, there are $\frac{U\ (U-1)}{2}$ models that are *trained* considering only two *classes* and perform a one-versus-one *classification* for the elements of the sample. This algorithm develops vectors that divide a hyperplane so that the *training sample* is split in *classes*. These vectors are then used to *classify* the *testing sample*. To identify the vectors, the primal problem is formulated (3.38).

$$\begin{aligned} \min\ & \frac{w^T w}{2} + C \sum_i \zeta_i \\ s.t.\ & y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i \\ & \zeta_i \geq 0\ \forall i \in 1, n \end{aligned} \tag{3.38}$$

In 3.38, x_i are the *training vectors* and $y \in \{-1; +1\}^n$ is a vector. The aim is to find the vector w and b so that the prediction of $(w^T\phi(x_i) + b)$ is corrected in the cases. In the primal problem, we are minimising the margin $\frac{w^T w}{2}$ and the penalties incurred for misclassification $C\sum_i \zeta_i$. The dual problem of (3.38) is (3.39).

$$\begin{aligned} \min \frac{\alpha^T Q \alpha}{2} - e^T\alpha \\ s.t.\ y^T\alpha = 0 \\ 0 \geq \zeta_i \geq C \forall i \in 1, n \end{aligned} \tag{3.39}$$

In 3.39, $Q = y_i y_j \phi(x_i)^T \phi(x_j)$, e is a vector of ones, and α_i are dual coefficient. The solution of the decision function for a sample x is $\sum y_i \alpha_i \phi(x_i)^T \phi(x_j) + b$.

- Stochastic gradient descent (SGD) is not itself a machine learning approach, but it is rather an optimisation technique that is used to minimise the loss function. By using gradient descent in machine learning, the misclassification is minimised and penalties are applied. SGD received attention thanks to its ability to deal efficiently with large-scale samples (Sra, Nowozin, and Wright 2012). SGD is not a specific machine learning model, but rather a methodology to find an approximate solution to a minimisation function of a problem. This methodology is applied to solve the log-likelihood function of the logistic regression. The stochastic gradient calculates the gradient of the function and uses it to identify the global minimum. In order to minimise the loss, the gradient of the loss function is calculated. Since the function might require a long time to be optimised, the system wanders randomly in order to reach the minimum, thus the name stochastic. This leads to the ability of the technique to deal with large-scale samples in a reasonable amount of time. In order to do so it moves randomly along the function hence the name stochastic.

- K-Nearest Neighbor (KNN) is a machine learning technique that can be applied to *classification* and regression problems. KNN is a variation of the Nearest Neighbor approach where the user can select the value of the k nearest neighbor to consider. During the *testing phase*, the components are assigned to a *class* based on the most common *class* of the k nearest elements. The euclidean distance is calculated to identify the nearest neighbors. In other words, if there are elements that are very similar to each other, they will be assigned to the same *class*. K-Nearest Neighbor relies on selecting k neighbors, with the number k provided by the

Table 3.6: Comparison of the machine learning algorithms

Machine Learning Approach and Algorithm	*Hyperparameters*	Classification Accuracy
Decision tree - CART	Maximum Number of Level of the Decision Tree, Minimum Size of the Sample to Create a Node, Minimum Size of the Sample to Create a Leaf Node	78.50%
Logistic Regression - L-BFGS		69.75%
Naive Bayes Algorithm		32.42%
Nearest Centroid Algorithm	Threshold for shrinking Centroids	4.60%
Random Forest Algorithm	Maximum Number of Level of the Decision Tree, Minimum Size of the Sample to Create a Node, Number of Decision Trees	78.35%
Support Vector Classifier	Maximum Number of Iterations, Tolerance, Penalty Parameter of the Error Term	57.84%
Stochastic Gradient Descent		66.15%
K-Nearest Neighbor Algorithm	Leaf Size of the Tree, Number of Neighbors to Use	77.83%

user, elements for the *classification* problem. In the *training phase*, the algorithm stores the *training sample*. In the *testing phase*, the algorithm identifies the k elements of the *training sample* that are near the element to *classify* and assigns their most common *class*. The algorithm uses the euclidean distance for the continuous *features* and the Hamming distance for the discrete ones. The former one is calculated as done for the centroids in the Nearest Centroid model, the latter one is the difference in symbols between the discrete *features*, i.e. text, of two elements.

In Table 3.6, we present the classification accuracy of the different algorithms together with the *hyperparameters* that are tuned for each algorithm.

Although the Classification And Regression Tree (CART) Algorithm is the one that outperforms the others, some other algorithms respond with an acceptable classification accuracy (K-Nearest Neighbor Algorithm and Random Forest Algorithm). On the other hand, these tools do not explain why a line feeding mode is assigned to each component, or, if they explain it, they lack clarity and conciseness. Decision tree learning provides clear reasons for assigning a line feeding mode, i.e. a *class*, to a

component. Because of this, it is superior to other algorithms.

3.4.5 Application

Unlike previous models that do not disclose the reasons for selecting a line feeding mode, a decision tree provides guidelines to practitioners who wish to better understand and optimise the ALFP, for a new assembly line or for rebalancing operations of an existing assembly line. Furthermore, a decision tree is simple and easy to interpret and allows the user to make good decisions in different industries.

An extract of a tree is depicted in Figure 3.9. The nodes show the Gini ratio, which measures the impurity of the node. As constraint (3.18), the root node of the tree *classifies* the components to *externally supplied sequencing* if the number of variants ($|S_k|$) is greater than or equal to the value of L_{seq}. Similarly, constraint (3.19) ensures that if a component's box has a volume higher than ω and more than one component, then *externally supplied sequencing* is applied. *Sequencing* is not assigned to components that have only one variant ($|S_k|$) as is enforced in constraint (3.22). Constraint (3.23) ensures that components that have a *box* with a volume that is lower than or equal to the recommended volume ($0.0208m^3$), they are *line stocked*.

It is important to remember that there are also some components that are wrongly *classified*. The radio dashboard is an example of a component that is wrongly *classified*. In this case, the line feeding mode assigned by the decision tree is *line stocking* at leaf node 5 instead of *stationary kit*. This can happen with multiple components, and this might lead to an infeasible instance. When this occurs, the repair approach assigns a line feeding mode to the component that violates the constraint. In order to reduce the cost deviation, this process is repeated. In this case, we consider the costs of the line feeding activities of each component. Since, for *kitted components*, there are also some costs that depends on the number of *kit containers* as considered in (3.2.5),(3.5.2), and (3.7), these costs are divided among all the components that are *kitted* and contribute to the implementation of the *kit container*. Then, we select the component with the highest costs and assign a line feeding mode.

Given that a component's attributes are known, the user can start from the root node and move along the branches until a leaf node — *class* and line feeding mode — is reached. The decision tree can be applied in a real manufacturing environment that is similar to the one we studied in this chapter. An example is described in Table 3.7. In this example, the line feeding modes are assigned to three new components.

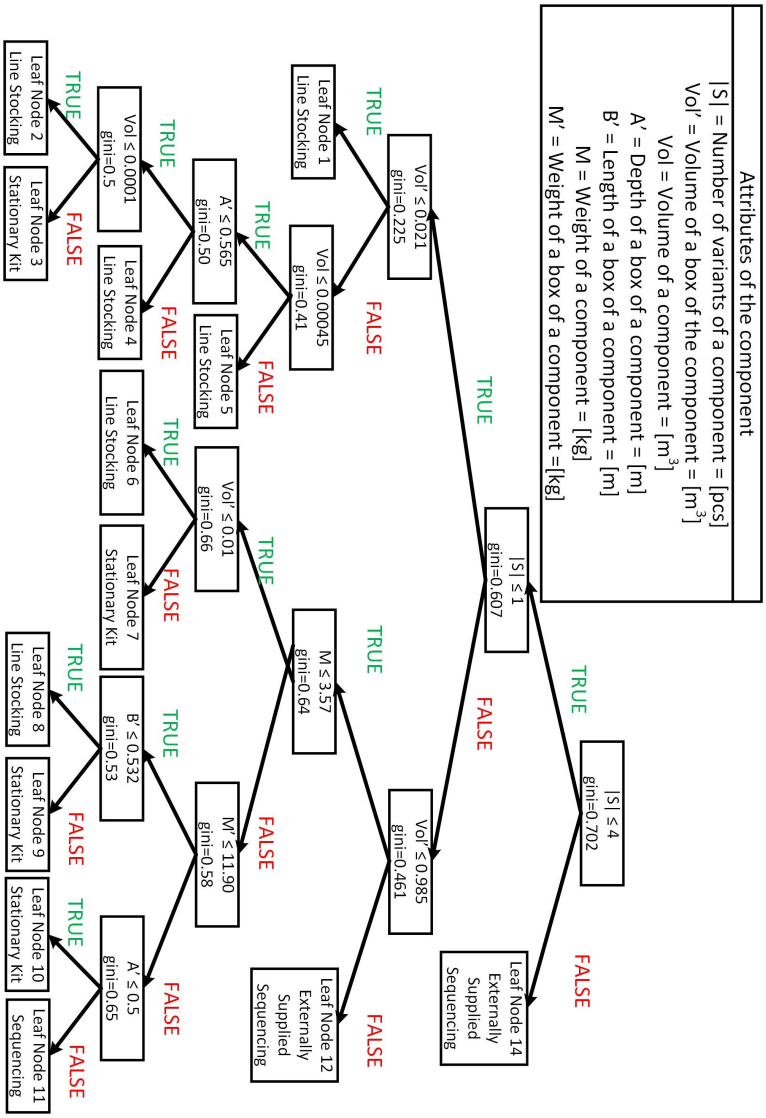


Figure 3.9: Decision tree example

3.4.6 Limitations

To explain the reasons for some of the decisions made, we would like to discuss the limitation of the algorithms and models. The repair approach that we developed assigns the optimal line feeding mode. Although this is an effective solution, it is not a common approach. Usually, a standard heuristic is compared to an optimisation model. In our case, due to the high number of constraints, this is not a viable solution. This is a limitation of our model. Future research should provide a solution to this problem. For instance, it should investigate the possibility of the tree to *classify* the components without leading to infeasible instances. This could be done by allowing the tree to consider multiple components, and if an instance becomes infeasible adjust the nodes that lead to the line feeding mode selection. Also, future research should focus on more elaborate techniques for the repair approach.

Another limitation consists of the parameter variation in the samples. Some of the parameters vary from one instance to the other such as the depth of a stationary kit container A^{sta} and the volume capacity of a stationary kit container Vol^{sta}. Other parameters, such as the interspace between two consecutive end products d and the labour cost per time unit C_0, remain the same for all the instances. The reason for this decision is that these parameters are usually standard in the manufacturing sector. Even in the literature (Limére, Van Landeghem, and Goetschalckx 2015; Limére et al. 2012; Müllerklein, Fontaine, and Ostermeier 2022; Sali and Sahin 2016; Sali, Sahin, and Patchong 2015; Schmid, Limére, and Raa 2021), these parameters have similar values. We selected these values according to the literature. For this reason, we do not vary these parameters among the instances. On this issue, we need also to be pragmatic to understand how to shape future research. If all the parameters in the model vary with wide ranges, the *classification accuracy* of the machine learning algorithm might decrease significantly.

An additional limitation is that the decision tree is developed using only attributes of the components. We purposely avoid using other parameters for the creation of the decision tree so that the *classification* depends only on the attributes of the components. This is to give managers insights based on the attributes of the components to make the best *classification* decision. Further, other parameters are randomly set in line with the ones found in the literature. Although the values and ranges are realistic, the randomness might lead to misleading assumptions. Though valuable insights can be obtained from a *classification* based on other parameters, it is more beneficial for managers to rely simply on the attributes of the components for this decision.

3.5 Conclusion

An existing optimisation model for the ALFP was enhanced. We synthetically generated the data to *train*, *validate*, and *test* the model. We used the CART algorithm to develop the decision tree. The tool was used for *classifying* components to a line feeding mode with a classification accuracy of 78.49%. We provided an example for the real-world application of the tool. To deal with the problem of infeasible *classifications*, we developed a repair approach that assigns the optimal line feeding modes. The cost deviation from the optimal solution is on average 0.38%.
In this chapter, we decided to consider the *stationary kit* rather than the *traveling kit* since the real data used in this work was provided by companies that implement the former. Future research could also investigate the implementation of the *traveling kit* instead of the *stationary kit*.

Table 3.7: Practical examples

Example of *Line Stocked Component*		Example of *Sequenced Component*	
Loudspeakers for cabin — Leaf Node 6		Silencer and tailpipe — Leaf Node 11	
Correctly *classified* ✓		Correctly *classified* ✓	
Attributes	Measures	Attributes	Measure
A	0.5 m	A	0.7 m
B	0.11 m	B	0.35 m
A'	0.3 m	A'	1.2 m
B'	0.2 m	B'	0.8 m
Vol	0.00016 m^3	Vol	0.085 m^3
Vol'	0.009 m^3	Vol'	0.64 m^3
M	0.3 kg	M	5 kg
M'	10 kg	M'	25 kg
$\|S\|$	2	|S|	4
p	32 pieces	p	4 pieces
Example of *Kitted Component*		Example of *Kitted Component*	
Wing Mirror — Leaf Node 7		Radio Dashboard — Leaf Node 5	
Correctly *classified* ✓		Wrongly *classified* ✗	
Attributes	Measures	Attributes	Measures
A	0.4 m	A	0.4 m
B	0.2 m	B	0.5 m
A'	1.2 m	A'	1.2 m
B'	0.8 m	B'	1.2 m
Vol	0.014 m^3	Vol	0.05 m^3
Vol'	0.58 m^3	Vol'	1 m^3
M	2.5 kg	M	3 kg
M'	43 kg	M'	32 kg
$\|S\|$	4	|S|	1
p	16 pieces	p	10 pieces

Chapter 4

Integrating Assembly Line Balancing and Feeding Decisions

Based on

Zangaro, F., S. Minner, and D. Battini. 2022. "The Multi-manned Joint Assembly Line Balancing and Feeding Problem." *International Journal of Production Research*. Advanced online publication. doi: 10.1080/00207543.2022.2103749.

The Joint Assembly Line Balancing and Feeding Problem (JALBFP) assigns a line feeding mode to each component (Assembly Line Feeding Problem) and each task to a workplace of a station (Assembly Line Balancing Problem). Current literature offers numerous optimisation models that solve these problems sequentially. However, only few optimisation models, provide a joint solution. To solve the JALBFP for a multi-manned assembly line, we propose a Mixed Integer Linear Programming (MILP) model and a heuristic that relies on the Adaptive Large Neighborhood Search (ALNS) framework by considering multiple workplaces per station and three different feeding policies: line stocking, travelling kitting and sequencing. The objective function minimises the cost of the whole assembly system which considers supermarket, transportation, assembly operations, and investment costs. Although the JALBFP requires higher computation times, it leads to a higher total cost reduction compared to the sequential approach. Through a numerical study, we validate the heuristic and find that the average deviation to the MILP model is around 1%. We also compare the solution of the JALBFP with that of the sequential approach and find an average total cost reduction of 10.1% and a maximum total cost reduction of 43.8%.

4.1 Introduction

In the automotive and machine tool manufacturing industry, more than one operator works in the same *station* to assemble a complex product made up of hundreds of different parts. These systems are called multi-manned assembly lines, since multiple workers (one for each *workplace* at each assembly *station*) are introduced to assemble the products, as opposed to a simple assembly line where only one operator works at each *station*. A *workplace* is a space in the *station* where assembly activities can be performed. When a new assembly line is constructed or rebalancing activities are performed, there is, at first, the need to identify the number of *stations* and *workplaces* where the assembly activities are performed. To do this, the Assembly Line Balancing Problem (ALBP) is solved. The optimal solution for the ALBP reduces the number of *stations* where the assembly activities are performed and minimises the investment cost of the assembly *stations* and the operating costs for the operators at the *workplaces*. In other words, it increases the space utilisation for the assembly activities but decreases the space that is available for storage at the Border of Line (BoL). After the numbers of *stations* and *workplaces* are selected, a line feeding mode must be assigned to each component that must be delivered to the assembly *stations*. At this stage, the number of *stations* and the space available at BoL cannot be increased and a high number of components must be kitted or sequenced so that the space at BoL is enough to host all components. The Assembly Line Feeding Problem (ALFP) assigns to each component a line feeding mode between line stocking, kitting, and sequencing. It minimises the sum of the investment costs for the line feeding activities, costs for the preparation activities for the line feeding modes, and transportation costs. The sequential implementation of the ALBP and the ALFP, as usually performed in a manufacturing environment, leads to the minimisation of the costs of the assembly line balancing activities at first, with a sequential increase of the line feeding activity costs. This leads to higher operational costs for the line feeding activities. The combination of the ALBP and the ALFP is the Joint Assembly Line Balancing and Feeding Problem (JALBFP). This problem identifies the number of *stations* and *workplaces* to perform the assembly activities and the line feeding mode of each component to achieve the global optimum of the sum of all investment and operating costs in a multi-manned assembly line.

Only a few models in the literature solve the JALBFP (Battini et al. 2016, 2017; Sternatz 2015). However, they focus the attention on direct and indirect line feeding modes without considering the sequencing policy (Calzavara et al. 2021), by

minimising total working times (Sternatz 2015) or the number of human operators involved in the assembly system (Battini et al. 2016, 2017) without taking into consideration the whole system costs. The major contribution of this chapter lies in the methodology, the optimisation model, and the algorithm that solves the JALBFP with a high level of detail. We present a Mixed Integer Linear Programming (MILP) model and a heuristic that solves the JALBFP for multi-manned assembly lines with a higher level of detail than previous models by considering different task times for each product model, multiple *workplaces* per *station* (Sternatz 2014), the kitting and sequencing line feeding mode, and by minimising the total system costs made up of supermarket, transportation, investment, and assembly costs. We use the Adaptive Large Neighborhood Search (ALNS) framework to solve large-scale instances of the JALBFP since the ALBP part of the problem is NP-hard. It determines which tasks should be performed at which *station* and how the components should be delivered to the assembly line. At the same time, it ensures that the precedence between the tasks is respected. This model is intended for a manufacturing environment where the space at BoL is scarce, e.g. in the automotive industry. Our MILP model solves the JALBFP with a cost objective function that considers multiple common operations that are performed in an assembly system rather than only the costs of the assembly operators or only some of the costs related to an assembly system. We validate the heuristic with the MILP model and find that the average deviation is about 1%. The JALBFP leads to average and maximum total cost reductions of 17.52% and 56.53% compared to the sequential approach. We also analyse the conditions under which total cost reduction is marginal (Wijnant, Schmid, and Limére 2018) and is higher in order to understand when it is more convenient to solve the JALBFP rather than the sequential approach. We use the heuristic to solve an instance of a real problem from a company that assembles mini-buses.
The remaining text is structured as follows: Section 4.2 describes the MILP model, Section 4.3 details the heuristic, Section 4.4 presents a numerical study, and Section 6 provides a summary and discussed the scope for future research.

4.2 Mathematical model

In this paragraph, we propose a new MILP model able to solve the JALBFP for multi-manned assembly lines and different part feeding policies. The objective function has been adapted from Sali and Sahin (2016).

4.2.1 Problem Description

In a multi-manned assembly line, a product moves from one *station* to the other. For each product, there are different product models N. In a multi-manned assembly line, each *station* has multiple *workplaces* W, which are locations in the assembly *station* where one operator performs assembly activities. Each task j, which is an operation performed by one operator, has a certain time duration known as task time T^{jn} that depends on the activities performed on a product model n. The task time is always lower than the *cycle time* C. There are precedences (π^j) that define the tasks i that should be performed before another task j. If a task i is a predecessor of a task j (π^j), it can only be initiated after the task j is completed even if it is performed in the same *station* but at a different *workplace*. We do not require additional constraints that enforce that two tasks should be simultaneously done by the operators or that require that two tasks must be performed one directly after the other one. Each task j requires a set of components ϕ^j that must be delivered to the same *station* where task j is performed. The attributes of the components (V^p, M^p, etc.) are characteristics of one single component that are equal for all its parts regardless of the variants, and the operators perform activities with a time duration t^t (see Table 5.1) to deliver the components. There is a cost C^0 for the assembly operators that work in the assembly *station* and C^l for the logistic operators in charge of the transportation and picking activities. The components must be delivered with a line feeding mode, i.e. line stocking, kitting, or sequencing. Line stocking consists of delivering multiple identical components in the same container as it is received at the warehouse. Sequencing involves the delivery of different variants, i.e. parts of the same component that might differ for some characteristics such as colour, texture, or material, in a single container. A (traveling) kit is a container with different components that are assembled on a single product that travels along the assembly line. While the tow trains that deliver line stocked components travel from the warehouse to BoL (D^{FB}), sequenced and kitted components are moved from the warehouse to the supermarket ($D^{F\upsilon}$) and from the supermarket to BoL ($D^{\upsilon B}$). Sequenced and line stocked components are placed at BoL while the kit container travels along the assembly line with the product. The tow trains that deliver sequenced and kitted components from the supermarket to BoL have a capacity in terms of containers Y. All assembly *stations* have a space at BoL with length L^s, where sequenced and line stocked components must be stored. The length L^2 is equal for all sequenced containers. There is a maximum space that can be occupied by a kit container S^3, and a kit container has a maximum volume

capacity V^3, a maximum weight capacity M^3, and a length L^3. There are fixed costs for implementing the traveling kit and sequencing C^f and variable costs for each component placed in the supermarket C^v. Each assembly *station* has a construction cost C^t. The whole assembly system consists of the assembly line, the supermarket and the activities that are performed to deliver the components. Figure 4.1 depicts the operations that are performed in an assembly line.

We use five decision variables for the ALBP: the binary assignment of task j to the *workplace* w at *station* s (x_{jws}), the time of the beginning of a task j (t_j), the binary implementation of *workplace* w at *station* s (y_{ws}), the order of implementing task i and task j (g_{ij}), and the binary implementation of *station* s (z_s). For the ALFP, we use four decision variables: the binary delivery of component p with line feeding mode m to *station* s (u_{pms}). Each component is delivered to the assembly line with a line feeding mode m. We index $m = 1$ for line stocked components, $m = 2$ for sequenced parts, and $m = 3$ for kitted components. q_2 and q_3 are binary decision variables that refer to the implementation of the sequencing and kitting line feeding modes while K is the number of kit containers (integer).

Table 4.1: Sets of the model, attributes of the components, and parameters of the model.

Sets of the model		
Not.	Definition	Set
π^j	Set of predecessors tasks of task j	—
ϕ^j	Set of parts p used for the task j	—
J	Set of tasks	$j, i = 1 \dots J'$
M	Set of supply modes	$m = 1 \dots M'$
N	Set of models of the final product	$n = 1 \dots N'$
P	Sets of components	$p = 1 \dots P'$
S	Set of assembly *stations*	$s, a = 1 \dots S'$
W	Set of *workplaces*	$w = 1 \dots W'$
W^j	Set of *workplaces* where task j can be performed	—
Attributes of the components		
Not.	Definition	Unit
ι^p	Consumption rate of a component p	%
λ^p	Number of sequenced carts for a component p	—
ν^p	Number of boxes consumed for a component p	—

σ^p	Capacity of a sequenced cart or a kit container for a for a component p	—
τ^p	Capacity of a tow train for a component p	—
Ξ^p	Length of the supermarket for one component p	m
c^p	Coefficient of BOM of a component p	pcs
F^p	Number of parts per box of a component p	$\frac{\text{pcs}}{\text{box}}$
L^p	Length of a box of a component p	m
M^p	Weight of a part of a component p	kg
r^p	Take rate of a component p	%
S^p	Surface space in the supermarket for a component p	m^2
Vol^p	Volume of a part of a component p	m^3
V^p	Number of variants of a component p	—

Parameters of the model

Not.	Definition	Unit
ϖ^j	Minimum length at BoL to stock all components p used for the task j	m
χ	Number of years of the annuity	years
ψ	Multiple delivery of the components (used to repair the solution of the ALBP for the ALFP)	—
ω	Present value factor of an annuity	—
C^0	Labour cost per time unit of an assembly operator	$\frac{€}{\text{s}}$
C^τ	Transportation costs	€
C	*Cycle time* or *takt time*	s
C^a	Costs for the assembly operations	€
C^f	Fixed costs of purchasing the traveling kit and the sequencing	$\frac{€}{\text{m}^2}$
C_I^i	Investment costs for the *stations*	€
C_{II}^i	Investment costs for the ALFP	€
C^l	Labour cost per time unit of a logistic operator	$\frac{€}{\text{s}}$
C^t	Costs for constructing an assembly *station*	€
C^u	Supermarket costs	€
C^v	Annual logistic labour cost of an operator	$\frac{€}{\text{year}}$
C^x	Variable costs of constructing a supermarket	$\frac{€}{\text{year m}^2}$

D^{Fv}	Distance between the warehouse and the supermarket	m
D^{FB}	Distance between the warehouse and BoL	m
D^{vB}	Distance between the supermarket and BoL	m
I	Interest rate	%
L^2	Length of a sequenced cart	m
L^3	Length of a kitting container	m
L^s	Length of an assembly *station*	m
M^3	Weight capacity of a kitting container	kg
S^3	Space for kit containers	m
T^{jn}	Task time of a task j for the product model n	s
V	Total annual production volume	$\frac{-}{\text{year}}$
v^0	Speed of an operator	$\frac{\text{m}}{\text{s}}$
V^3	Volume capacity of a kitting container	m^3
v^t	Speed of the tow train	$\frac{\text{m}}{\text{s}}$
Y	Capacity of a tow train for kitted and sequence containers from the supermarket to BoL	—
	Time needed for	
t^{ak}	A single movement of picking variants during the preparation in a kitting mode.	s
t^{aks}	Additional operation of loading and unloading a container of a component in kitting and sequencing mode.	s
t^{as}	A single movement of picking and loading variants during the preparation in a sequencing mode.	s
t^{gl}	A single movement of grasping boxesz and loading and unloading them.	s
t^{luk}	A single movement of loading or unloading a kit container from the train.	s
t^{lus}	A single movement of loading or unloading a sequenced part from the train.	s

We use five decision variables for the ALBP: the binary assignment of task j to the *workplace* w at *station* s (x_{jws}), the time of the beginning of a task j (t_j), the binary

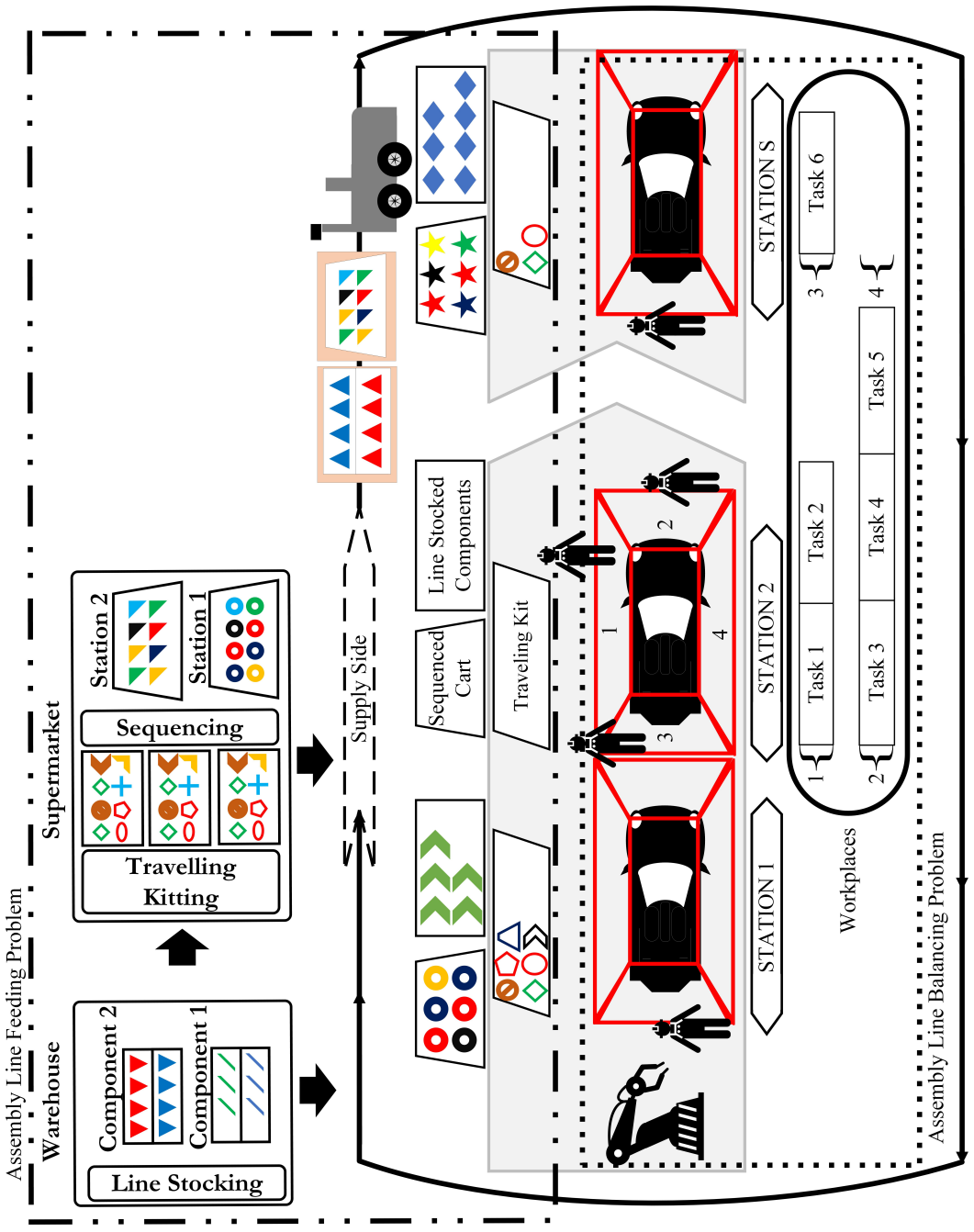


Figure 4.1: Graphical representation of a multi-manned assembly system with different part feeding modes implemented in each *station*.

implementation of *workplace* w at *station* s (y_{ws}), the order of implementing task i and task j (g_{ij}), and the binary implementation of *station* s (z_s). For the ALFP, we use four decision variables: the binary delivery of component p with line feeding mode m to *station* s (u_{pms}). Each component is delivered to the assembly line with a line feeding mode m. We index $m = 1$ for line stocked components, $m = 2$ for sequenced parts, and $m = 3$ for kitted components. q_2 and q_3 are binary decision variables that refer to the implementation of the sequencing and kitting line feeding modes while K is the number of kit containers (integer).

4.2.2 Objective Function

The objective function minimises the total costs of the assembly system made up of investment and operational costs. The operational costs in (4.1.2) are the annual costs incurred over the entire time of the assembly line utilisation, while the investment costs in (4.1.1) are relevant at the moment of construction. The factor ω is used for calculating the present value of the future expenses.

$$\min C_{tot} = \underbrace{C_I^i + C_{II}^i}_{(4.1.1)} + \omega(\underbrace{C^u + C^\tau + C^a}_{(4.1.2)}) \tag{4.1}$$

Assembly Operations (C^a) : There is a cost for the operators that perform the assembly activities in each *workplace*. In (4.2), we multiply the number of implemented *workplaces* with the labour assembly costs (C^0), the *cycle time* (C), and the total annual production volume(V).

$$C^a = C^0 C V \sum_{w \in W} \sum_{s \in S} y_{ws} \tag{4.2}$$

Assembly Stations (C_I^i) : Each *station* has an investment cost that is not dependent on the number of *workplaces*. These are the costs for all assembly *stations*. In (4.3), we multiply the total number of implemented *stations* with the costs of an assembly *station* C^t.

$$C_I^i = C^t \sum_{s \in S} z_s \tag{4.3}$$

Supermarket Costs (C^u) : Some activities are performed to place each component in the right container to be delivered to the assembly line. To prepare the kitted and sequenced containers, a roundtrip must be made with the tow train to collect all components. For the sequenced components in (4.4.1), the number of sequenced containers (λ^p) is calculated by dividing the consumption rate (ι^p), which is the product of the take rate (r^p) and the coefficient of Bill Of Material (BOM) (c^p), by the capacity of a tow train for a component p (σ^p). This number is multiplied by the length of the supermarket of one component p (Ξ^p). The sum over all components P and all *stations* S, which is equal to the number of tow trains, is divided by the speed of the operator (v^0). Similarly, for kitting components in (4.4.2), the length of the supermarket is multiplied by one component p (Ξ^p).
To collect both the sequenced and the kitted components, the consumption rate (ι^p) in (4.4.3) and (4.4.4) is used. However, we distinguish between the picking time for sequenced components (t^{as}) and the picking time for kitted components (t^{ak}).
Subsequently, some additional picking activities must be performed due to the intermediate supermarket where the components are placed. For both the kitted and the sequenced components in (4.4.5), the time needed for the additional activities (t^{aks}) is multiplied by the number of boxes (ν^p). The number of boxes is equal to the consumption rate (ι^p) divided by the fill level of a component (F^p).
Eventually, the *containers* must be loaded onto the tow train. This calculation is done for line stocked components in (4.4.6), but, in this case, a time (t^{gl}) is used. For sequenced components in (4.4.7), the calculation is similar to what is done in (4.4.6). However, for these components, the time for loading and unloading sequenced parts t^{lus} is used. For kitted components in (4.4.8), the time t^{luk} is multiplied by the number of kit containers K.
In all these computations, the annual costs for the logistic operations C^v are used, which is the labour logistic cost (C^l) multiplied by the total annual production volume (V).
There is a cost for the floor space occupied for the storage of the kitted and sequenced components that are placed in the supermarket. These costs are calculated by multiplying the costs of a square metre (C^x) with the number of variants V^p for the surface space S^p in (4.4.9).

$$
\begin{aligned}
C^u = \frac{C^v}{v^0}\Bigg\{ \sum_{p\in P}\sum_{s\in S}\Big[\underbrace{\lambda^p \Xi^p u_{p2s}}_{(4.4.1)} + \underbrace{\Xi^p u_{p3s}}_{(4.4.2)} \Big] \Bigg\} + \\
C^v \Bigg\{ \sum_{p\in P}\sum_{s\in S}\Big[\underbrace{t^{as}\iota^p u_{p2s}}_{(4.4.3)} + \underbrace{t^{ak}\iota^p u_{p3s}}_{(4.4.4)} + \\
\underbrace{t^{aks}\nu^p(u_{p2s}+u_{p3s})}_{(4.4.5)} + \underbrace{t^{gl}\nu^p u_{p1s}}_{(4.4.6)} + \\
\underbrace{t^{lus}\lambda^p u_{p2s}}_{(4.4.7)}\Big] + \underbrace{K t^{luk}}_{(4.4.8)} \Bigg\} + \underbrace{C^x \sum_{p\in P}\sum_{s\in S} S^p V^p (u_{p2s}+u_{p3s})}_{(4.4.9)}
\end{aligned}
\tag{4.4}
$$

Transportation Costs (C^τ) : Transportation activities are performed to deliver the components to the assembly line for all three line feeding modes (Battini et al. 2015). While line stocked components are delivered directly to BoL from the warehouse, kitting and sequenced components go through the supermarket. First, the kitted and sequenced components are transported from the warehouse to the supermarket as in (4.5.1). The calculation is similar to that in (4.4.5), and the number is divided by the capacity of a tow train used for this operation (τ^p). The time duration of the trip from the warehouse to the supermarket is calculated by using the distance D^{Fv}.

After the kits and the sequenced components are transported to the supermarket, they must be moved to BoL. In (4.5.2), the number of kit containers is multiplied by the distance travelled D^{vB} divided by the capacity in terms of containers of a tow train Y. In (4.5.4), we use the number of kit containers K multiplied by the distance between the supermarket and BoL (D^{vB}) and divided by the capacity in terms of containers of a tow train (Y). Line stocked components must be delivered directly from the warehouse to BoL. Similar to (4.5.1), we calculate the milk-runs for the *boxes* of line stocked components in (4.5.3) with the distance between the warehouse and BoL (D^{FB}).

In all these computations, we use the annual logistic operation costs C^v divided by the speed of the tow train (v^t).

$$C^{\tau} = \frac{C^{v}}{v^{t}} \Bigg\{ \sum_{p \in P} \sum_{s \in S} \Big[\underbrace{D^{F} v \frac{\nu^{p}}{\tau^{p}} (u_{p2s} + u_{p3s})}_{(4.5.1)} + \underbrace{\frac{D^{vB}}{Y} \sum_{p \in P} \sum_{s \in S} \lambda^{p} u_{p2s}}_{(4.5.2)} + \underbrace{D^{F} B \sum_{p \in P} \sum_{s \in S} \frac{\nu^{p}}{\tau^{p}} u_{p1s}}_{(4.5.3)} \Big] + \underbrace{\frac{D^{vB}}{Y} K}_{(4.5.4)} \Bigg\} \tag{4.5}$$

Investment Costs for the ALFP (C^{i}_{II}) : Investment costs for the supermarket's equipment are necessary for the kitting and sequencing activities. The fixed cost (C^{f}) is multiplied by the decision variables q_2 and q_3.

$$C^{i}_{II} = C^{f}(q_2 + q_3) \tag{4.6}$$

4.2.3 Constraints

There are three types of constraints: *Assembly Line Balancing constraints*, *Assembly Line Feeding constraints*, and *variable domains*.

Assembly Line Balancing Constraints:

$$\sum_{s \in S} \sum_{w \in W} x_{jws} = 1, \quad \forall j \in J \tag{4.7}$$

$$\sum_{j \in J} T^{jn} x_{jws} \leq C y_{ws}, \quad \forall w \in W, s \in S, \; n \in N \tag{4.8}$$

$$y_{ws} \leq z_{s}, \quad \forall w \in W, s \in S \tag{4.9}$$

$$\sum_{s \in S} s \sum_{w \in W} x_{iws} \leq \sum_{s \in S} s \sum_{w \in W} x_{jws}, \quad \forall j \in J, i \in \pi^{j} \tag{4.10}$$

$$t_j \geq t_i + T^{in} - M(1 - \sum_{w \in W} x_{jws}) - M(1 - \sum_{w \in W} x_{iws}), \quad \forall j \in J, i \in \pi^{j}, s \in S, n \in N \tag{4.11}$$

$$t_j + T^{jn} \leq C, \quad \forall j \in J, n \in N \tag{4.12}$$

$$t_j \geq t_i + T^{in} - M(1 - x_{iws}) - M(1 - x_{jws}) - M(1 - g_{ij}), \quad \forall j, i \in J, i \neq j, \; w \in W, s \in S, \; n \in N \tag{4.13}$$

$$t_i \geq t_j + T^{jn} - M(1 - x_{iws}) - M(1 - x_{jws}) - Mg_{ji}, \quad \forall j, i \in J, i \neq j, \; w \in W, s \in S, \; n \in N \tag{4.14}$$

Constraints (4.7) enforce that each task j is performed in one assembly *station* s. Constraints (4.8) guarantee that the sum of the duration of all tasks j for all models n that are assigned to a *workplace* w at a *station* s must be lower than the *cycle time*. These constraints also ensure that a task j can only be performed in a *workplace* w in a *station* s, and only if that *workplace* w is implemented in *station* s. Constraints (4.9) define that implementation of a *workplace* w at a *station* s can only take place if that *station* s is implemented.

If a task i is a predecessor of a task j (π^j), it can only be initiated after task j is completed even if it is performed in the same *station* but at a different *workplace*. For simplicity, we do not require additional constraints that enforce that two tasks should be simultaneously done by the operators or that require that two tasks must be performed one directly after the other one.

For the precedences between the tasks, we consider the starting time of the tasks that have a precedence and are performed in the same station (Çil and Kizilay 2020; Naderi, Azab, and Borooshan 2019; Roshani and Nezami 2017; Sahin and Kellegöz 2019). If a task i should be performed before another task j, then the task i should be performed in a previous or the same *station* of task j. To do this, constraints (4.10) ensure that if one task i is a predecessor of another task j, then task i must be performed at a previous or the same *station* where task j is performed. Additionally, a set of constraints known as sequencing constraints that are described in Michels et al. (2019) must be introduced to ensure that the scheduling is respected if tasks are performed at different workplaces. These constraints ensure that in a *station* the time to perform the assembly tasks is lower than the cycle time also by considering the precedence between the tasks that are performed in a different *workplace*. To do this, constraints (4.11) are Big M constraints and ensure that the starting time t_j of each tasks j occurs after the completion of the task i predecessor of j (π^j). The Big M elements enforce constraints (4.11) only if the

two tasks i and j are performed in the same *station*. Constraints (4.12) ensure that the tasks are completed before cycle time C.
Two more sets of constraints must be introduced to satisfy the precedence of the tasks (Michels et al. 2019; Naderi, Azab, and Borooshan 2019; Sahin and Kellegöz 2019). Constraints (4.13) and (4.14) control the time of the beginning of the tasks that do not have precedence restrictions. If task i is performed before task j, g_{ij} is equal to 1, and 0 otherwise. Constraints (4.13) ensure that if two tasks are assigned to the same station, then task j is performed only after task i is completed. Constraints (4.14) ensure the same aspect when task i is performed after task j, i.e. g_{ij} is equal 0. If two tasks are performed at the same station, one of the two must be performed before the other. The binary decision variables g_{ij} ensure also that two tasks cannot be performed simultaneously.

Assembly Line Feeding Constraints:

$$\sum_{m \in M} u_{pms} = \sum_{j \in J} \sum_{w \in W} x_{jws}, \quad \forall p \in \phi^j, s \in S \tag{4.15}$$

$$\sum_{p \in P} \Xi^p u_{p1s} + \sum_{p \in P} L^2 u_{p2s} \leq L^s, \quad \forall s \in S \tag{4.16}$$

$$KL^3 \leq S^3 \tag{4.17}$$

$$\sum_{p \in P} \sum_{s \in S} c^p u_{p3s} M^p \leq KM^3 \tag{4.18}$$

$$\sum_{p \in P} \sum_{s \in S} c^p u_{p3s} V^p \leq KV^3 \tag{4.19}$$

$$q_m \geq u_{pms}, \quad \forall s \in S, p \in P, m \in \{2; 3\} \tag{4.20}$$

$$\sum_{s \in S} \sum_{m \in M} u_{pms} = 1, \quad \forall p \in P \tag{4.21}$$

Constraints (4.15) force a component p supplied with any line feeding mode m to be delivered to an assembly *station* s where task j is performed. Constraints (4.16) ensure that a *station* s must be long enough to provide space for all line stocked components and all sequenced parts. Constraint (4.17) guarantees that all kitting containers can be stored in the kitting area. Constraints (4.18) and (4.19) set the number of kit containers K for delivery. Constraints (4.20) enforce that q_2 and q_3 are equal to 1 if the sequencing and kitting line feeding modes are used. Constraints (4.21) ensure that the components are delivered to only one *station* with only one line feeding mode.

Variable Domains:

Aside from the number of kit containers K which is an integer, all other decision variables are binary.

$$K \in \mathbb{N}, u_{pms}, x_{jws}, y_{ws}, z_s, q_2, q_3, g_{ij} \in \{0,1\}, t_j \in \mathbb{R}^+ \quad \forall p \in P, m \in M, s \in S, w \in W, j \in J, i \in J \tag{4.22}$$

4.3 Adaptive Large Neighborhood Search Heuristic

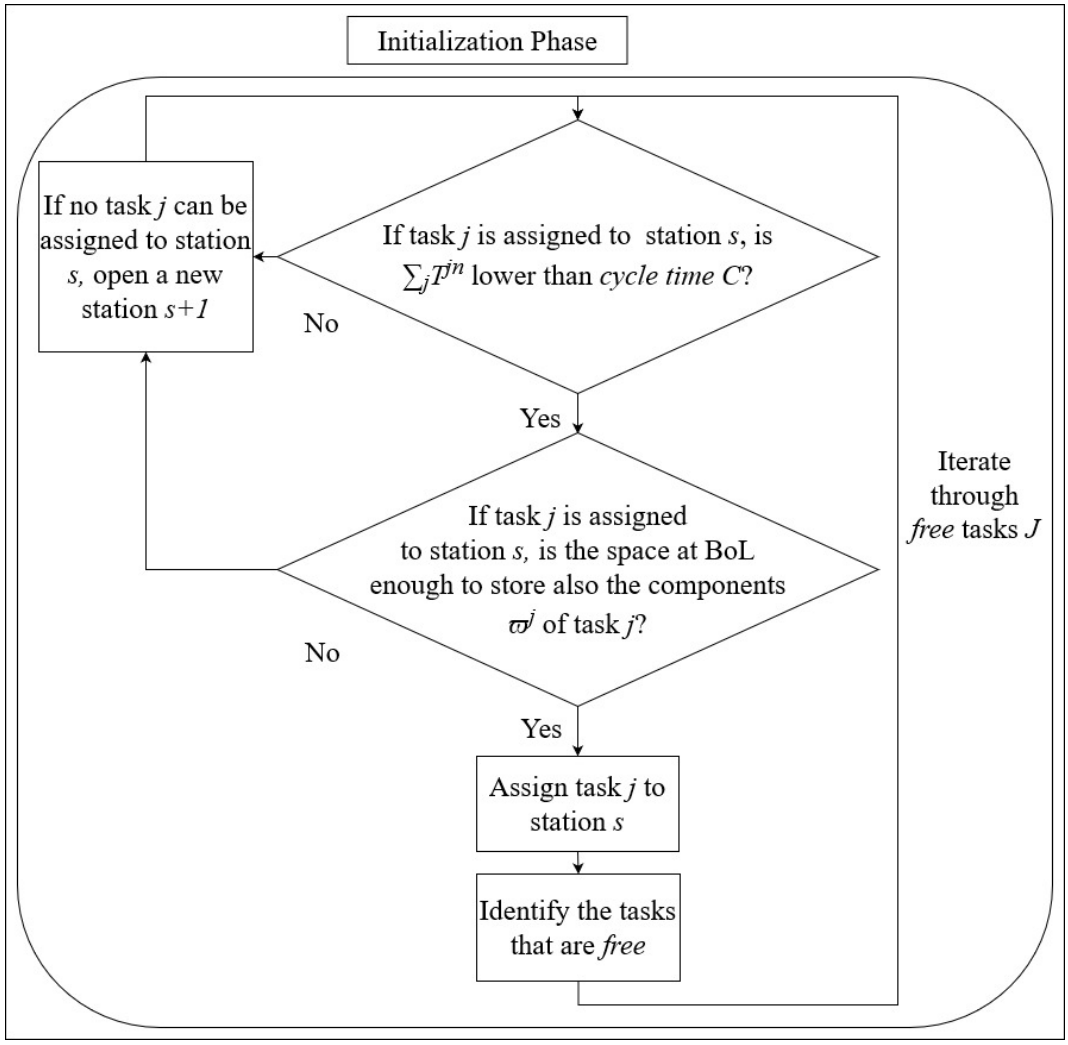
We now describe the heuristic for solving large-scale instances based on the ALNS framework. Large Neighborhood Search (LNS) is first presented by Shaw (1998). Its main idea is to progressively improve an initial solution, generated from an *initialisation phase*, by repeatedly removing elements from the solution (removal operators) and inserting them back into it (insertion operators). Ropke and Pisinger (2006) extend LNS with ALNS for the pickup and delivery problem with time windows, which provides multiple removal and insertion operators that are selected based on their achieved performances. Figure 5.3 shows the flowchart of the *initialisation phase* and the algorithm of the heuristic approach.

The ALNS algorithm is divided into three elements:

1. *The initialisation phase* develops an initial feasible solution. We define as *free tasks* the ones that have no predecessors or the predecessors of which are already assigned to other *stations*. At each *station*, we assign the *free tasks* to a *station* if the *cycle time* C and the space at BoL (ϖ^j) allow it. We also ensure that any task begins after its predecessors that are assigned to the same *station*. If there is no *free task* that can be assigned to a *station*, a new *station* is opened, and the step continues with the remaining *free tasks* (see Figure 5.3).

2. *The improvement phase* enhances the initial solution that is generated. The removal operator eliminates some tasks from the solution, while the insertion operators introduce them back in the solution in a different position. There are multiple removal operators:

   - One random task is removed from the solution.
   - Multiple random tasks are removed from the solution.



initialisation phase;
store the current solution as the minimum solution;
while *stopping criterion is TRUE* **do**
    use a removal operator;
    use a repair operator;
    calculate C'_{tot} of the current solution;
    if $C'_{tot} < C_{tot}$ **then**
        store the current solution as the minimum solution;

Figure 4.2: Flow chart of the *initialisation phase* and algorithm of the heuristic.

- One random *workplace* and all tasks that are assigned to this *workplace* are removed from the solution.
- Multiple random *workplaces* and all tasks that are assigned to these *workplaces* are removed from the solution.
- One random *station* and all tasks that are assigned to this *station* are removed from the solution.
- Two random *stations* and all tasks that are assigned to these *stations* are removed from the solution.

Typically, other removal operators (worst-distance, worst-time, or proximity-based removal) that rely on the distances between the *stations* are implemented in ALNS for a Vehicle Routing Problem (VRP). However, these removal operators cannot be implemented for the JALBFP. There are four main insertion operators:

- The removed element is introduced in a random *station* that is already open.
- The removed element is introduced in a new *station*.
- The removed element is exchanged with another element of the solution.
- The removed element is introduced in the best *workplace* based on the partial solution.

The introduction or exchange with another task or *station* is performed if it does not violate the precedence constraints. For instance, a task i at *station* s_i that is a predecessor of another task j at *station* s_j (π^j) cannot be inserted after *station* s_j. The *stopping criterion* requires at least η_1 iterations and η_1' iterations after the last feasible improved solution is found. During the *improvement phase*, the operator that is most effective in decreasing the objective value is selected. In this phase, there is, at first, a warm-up period of ψ iterations during which any random removal and insertion operator can be selected. After the warm-up period, i.e. the first ψ iterations, the algorithm selects at each iteration one removal operator based on the success rate of each removal and insertion operator. To select the operator that is used, we assign weights to the different operators and use a roulette wheel selection principle. The weight w_i is the number of successes of the operator i divided by the total. The ALFP part of the problem, i.e. the assignment of the components to a line feeding mode, is performed with the optimisation model. During the improvement process, we restrict the solution to

nodes that yield an objective value better than the one already obtained. We do not pursue nodes that yield a worse objective value than the ones already obtained because this solution will not be accepted.

3. *The acceptance method* is the condition used to evaluate the changes made by the remove-insert operators. To identify a new solution, the algorithm accepts a feasible solution with an objective value that is lower than the previous one.

4.4 Numerical study

4.4.1 Performance Measures

To compare the quality of the results for the sequential implementation of the ALBP and ALFP, and the JALBFP, we define five performance measures:

- The *computation time* necessary to solve the MILP with a commercial solver and the ALNS heuristic.
- The *worker density*, which is the number of operators in each *station*.
- The *component density*, which is the number of components delivered to each *station*. The number of delivered components can be divided into the three line feeding modes used for the delivery: line stocking, kitting, and sequencing.
- The *idle time*, which is the percentage of *cycle time* during which the operator is idle and does not perform productive activities. Such unproductive activities should be taken into consideration.
- The *total cost reduction* achieved through the implementation of the JALBFP rather than the sequential approach.

Worker density and *component density* are measures that are taken into consideration by managers and practitioners in the planning phase of an assembly line. Managers consider the *worker density* and the *component density* as a comparison measure between assembly lines.

4.4.2 Data Generation

We obtain the task times and the precedence diagrams from instances in Otto, Otto, and Scholl (2013) and use 'less tricky' and 'tricky' instances that contain 20 tasks J' (small size) and 'less tricky' instances that contain 50 and 100 tasks J' (medium and large scale). A limitation of the solution method proposed in this chapter is that it is not able to solve instances that are classified as 'tricky', 'extremely tricky', and 'open' or instances that have 1 000 tasks. Indeed, some of the instances with 100 and 1 000 tasks J, especially those marked as 'tricky', 'extremely tricky', and 'open', remain difficult to solve for the ALBP or remain unsolved (Akpinar, Elmi, and Bektaş 2017; Álvarez-Miranda, Chace, and Pereira 2020; Cerqueus and Delorme 2019; Li, Kucukkoc, and Tang 2020). These instances contain the task times and the precedence diagram for the assembly operations of products with one model. The task times and the *cycle times* in these instances are reported in time unit (TU) with 10 TU being equal to 1 second. Because we consider N' models of a product, we arrange the task times in the same way as N' instances from Otto, Otto, and Scholl (2013). The task times T^{jn} of a model n is the task time of task j of instance n. The instances of Otto, Otto, and Scholl (2013) also provide the *cycle time* C. We select the precedence diagram from the first instance of a group of N' instances. For the set of *workplaces* W^j where the task j can be performed, we assign a random number of *workplaces* to those tasks that are at the first level on the precedence diagram with the uniform distribution. Then, we assign one random *workplace* of the predecessor π^j to each remaining task j. Furthermore, we determine the number of components P' used in one task j based on the task time $\overline{T^{jn}}$.

For the attributes of the components, we implement an approach that is based on the resample and replacement principle (Efron and Tibshirani 1993) and use data that is provided by multiple companies and developed based on the authors' experience. Given a population F of dimension α and size γ, a random sample $\mu_1, \mu_2, \ldots, \mu_n$ is obtained. This is achieved by a selection of n random integer numbers between 1 and γ, each with a probability of $\frac{1}{\gamma}$. Because this is an approach with replacement, the integer numbers can be repeated. A repetition will lead to $\mu_1 = F_1, \mu_2 = F_2, \ldots, \mu_n = F_n$. Each member of the new sample μ_n will have the same dimension α as the population F. This approach allows the generation of scenarios even if the observed data's statistical distribution cannot be estimated, is unknown, or lacks accuracy. Table 4.2 provides the values, ranges, and sets of the parameters of the model.

The remaining parameters of the model in Table 4.2 are set following the authors'

experience. As described in Sternatz (2015), we alter the distances between the warehouse, supermarket, and BoL (D^{FB}, D^{Fv}, and D^{vB}).
The remaining parameters of the model in Table 4.2 are set following the authors' experience. As described in Sternatz (2015), we alter the distances between the warehouse, supermarket, and BoL (D^{FB}, D^{Fv}, and D^{vB}).

Table 4.2: Sets, ranges, and values of the sets of the model; parameters of the model; and attributes of the components.

Sets of the model		
Not.	Set, Range, or Value	Unit
J'	$\{20; 100\}$	—
M	$[1, 3]$	—
N'	5	—
W'	4	—
Attributes of the components		
Not.	Range	Unit
ι^p	*Calculated* $(r^p c^p)$	%
λ^p	*Calculated* $(\frac{r^p c^p}{\sigma^p} = \frac{\iota^p}{\sigma^p})$	—
ν^p	*Calculated* $(\frac{r^p c^p}{F^p} = \frac{\iota^p}{F^p})$	—
σ^p	$[1, 280]$	—
Ξ^p	*Calculated* $(L^p V^p)$	m
τ^p	$[2, 50]$	—
c^p	$[1, 20]$	pcs
F^p	$[1, 12000]$	$\frac{\text{pcs}}{\text{box}}$
L^p	$[0.05, 4.5]$	m
M^p	$[0.001, 150]$	kg
r^p	$[0.0001, 1]$	%
S^p	$[2, 14]$	m^2
V^p	$[1, 14]$	—
Vol^p	$[0.001, 8]$	m^3
Parameters of the model		
Not.	Set, Range, or Value	Unit
ϖ^j	$[0, 6.5]$	m
χ	5	years

ω	*Calculated* ($\sum_{\rho=1}^{\chi} \frac{1}{(1+I)^{\rho}} = 4.33$)	—
C^0	0.014	$\frac{€}{\mathrm{s}}$
C	*Obtained from* Otto, Otto, and Scholl (2013)	s
C^f	10 000	$\frac{€}{\mathrm{m}^2}$
C^l	0.010	$\frac{€}{\mathrm{s}}$
C^t	400 000	€
C^v	*Calculated* ($C^l V$)	$\frac{€}{\mathrm{year}}$
C^x	150	$\frac{€}{\mathrm{year}\,\mathrm{m}^2}$
D^{Fv}	$\{400; 880; 1700\}$	m
D^{FB}	$\{800; 1700; 3400\}$	m
D^{vB}	$\{300; 750; 1500\}$	m
I	0.05	%
L^2	1	m
L^3	3.5	m
L^s	8	m
M^3	25	kg
S^3	7	m
t^{ak}	$[28, 63]$	s
t^{aks}	81.1	s
t^{as}	$[28, 63]$	s
t^{gl}	16.5	s
T^{jn}	*Obtained from* Otto, Otto, and Scholl (2013)	s
t^{luk}	81.1	s
t^{lus}	16.5	s
v^0	0.9	$\frac{\mathrm{m}}{\mathrm{s}}$
V^3	3	m^3
V	140 000	$\frac{-}{\mathrm{year}}$
v^t	1.94	$\frac{\mathrm{m}}{\mathrm{s}}$
Y	5	—

4.4.3 Real Application in the Automotive Sector

We solve a real instance provided by a company that assembles mini-buses to show how the heuristic can be implemented in a real-world environment. This instance includes 80 tasks and 910 components that are delivered to the assembly line for a chassis of a

mini-bus. The whole assembly process is divided in parts with 100 tasks or less.

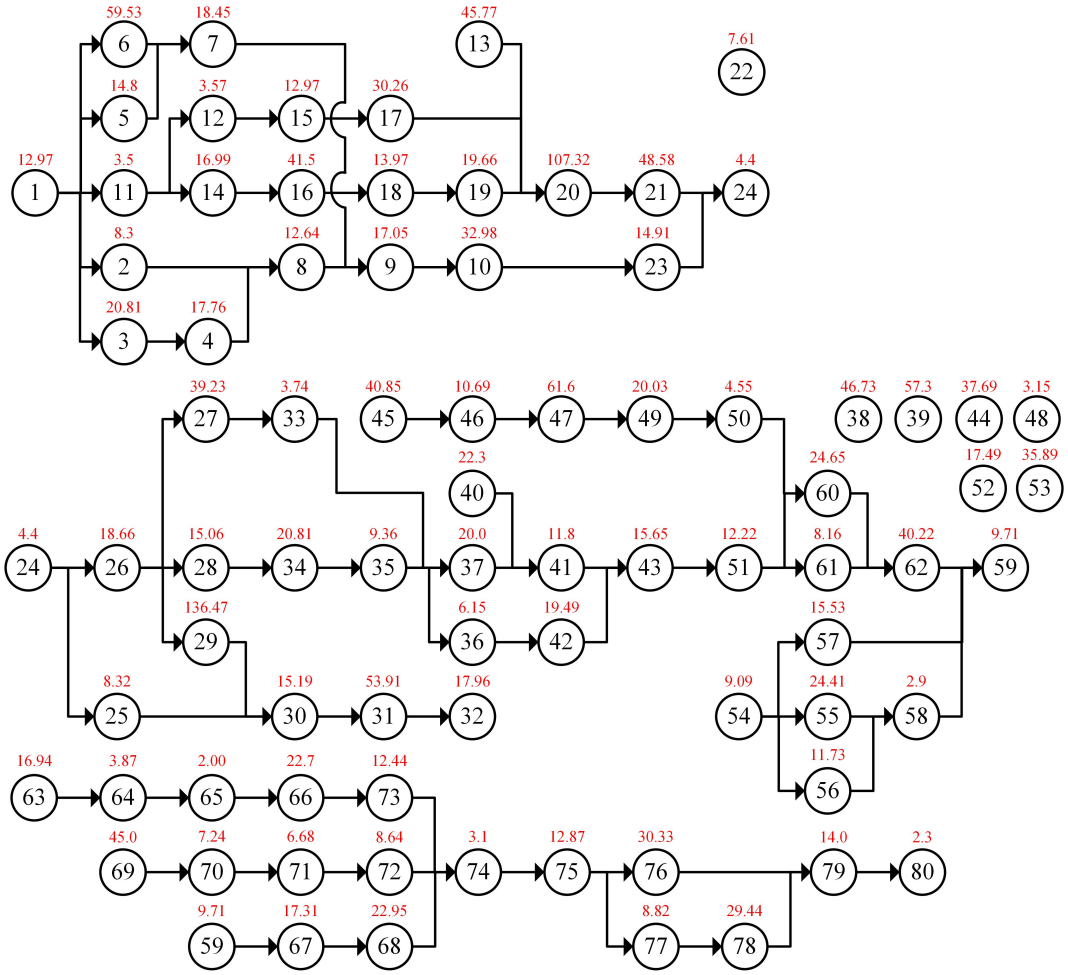


Figure 4.3: Precedence diagram of the task for the assembly of a mini-bus. The circles represent the tasks and the average execution time is written in red.

One might argue that the whole assembly system that is part of a manufacturing environment or an automotive environment is composed of some hundreds of different tasks and thousands of components and that considering only 80 tasks is not a realistic problem.

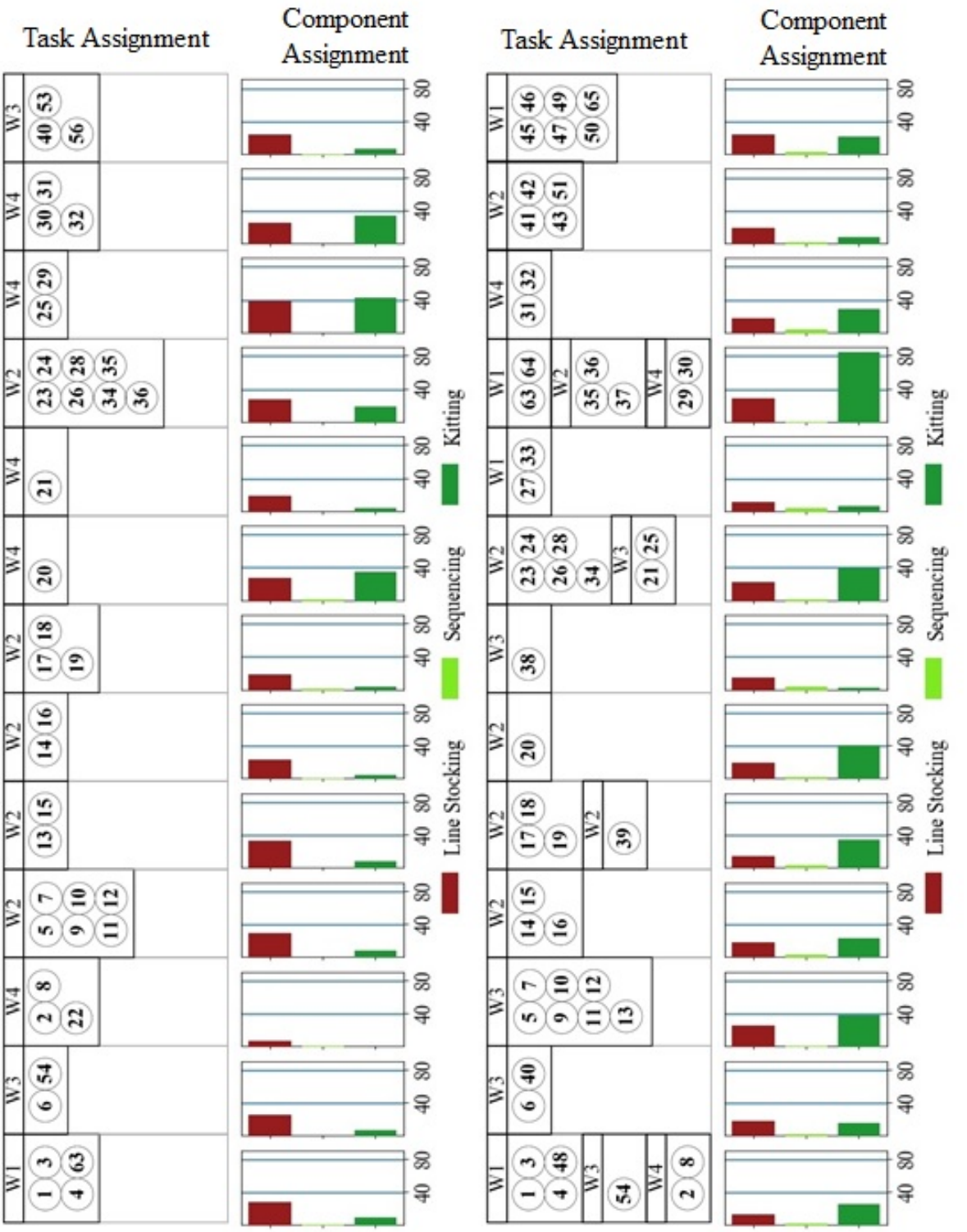


Figure 4.4: Schema of the solution of the JALBFP and the sequential approach for the first part of the assembly line. The line feeding mode encoded in red requires only minimal preparation activities. The line feeding modes encoded in green (sequencing and kitting) require more extensive preparation activities at the supermarket.

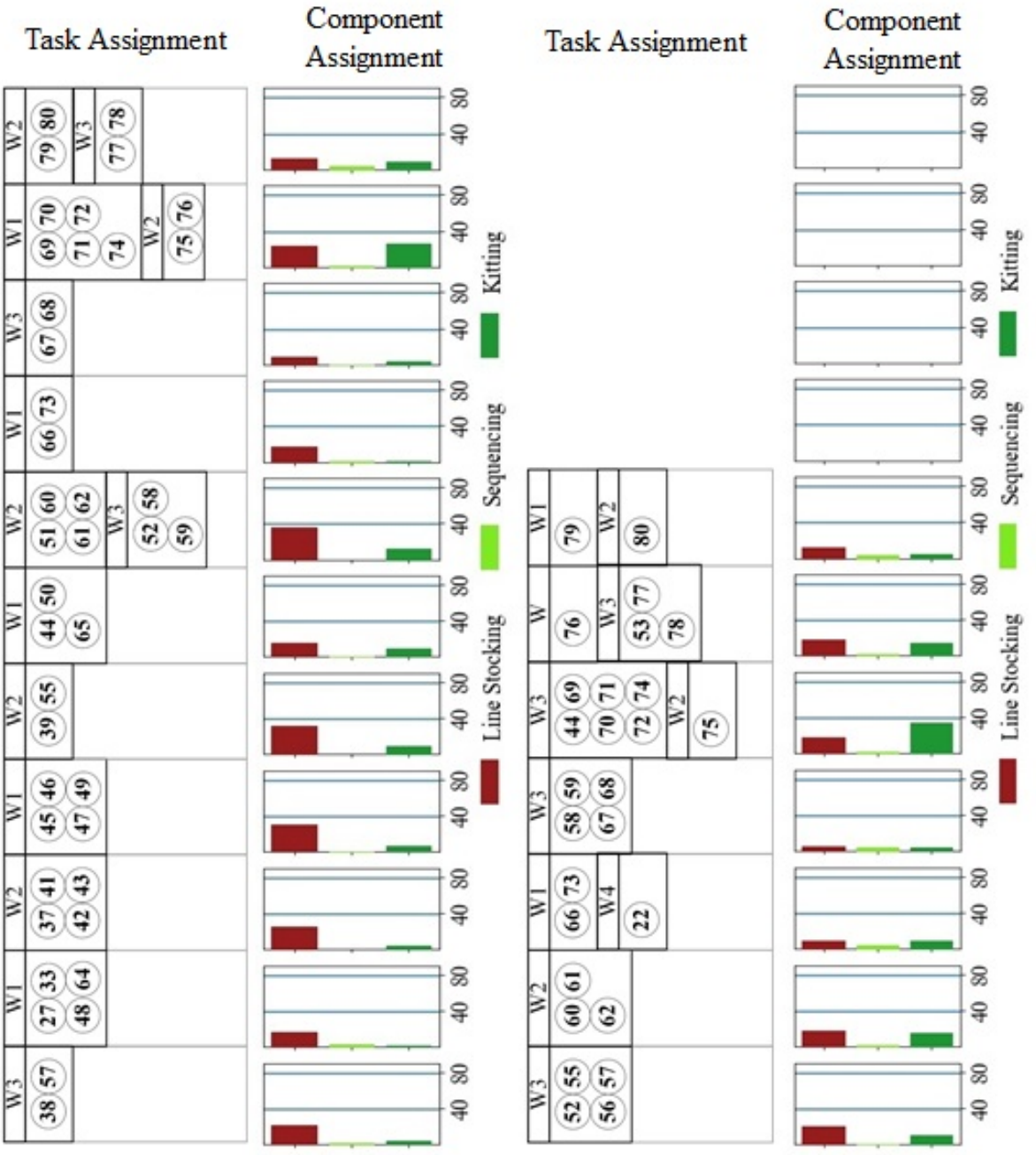


Figure 4.5: Schema of the solution of the JALBFP and the sequential approach for the second part of the assembly line. The line feeding mode encoded in red requires only minimal preparation activities. The line feeding modes encoded in green (sequencing and kitting) require more extensive preparation activities at the supermarket.

Although the whole assembly system is indeed made up of a hundred of different tasks, it is usually divided into multiple sections, each of them consisting of around one hundred tasks, a few hundred components, and few dozens of *stations* (Sternatz 2015). In other words, the whole assembly system is divided in multiple sections each one consisting of some tasks, their precedences, and their components. Indeed, due to the precedences between the tasks it is not necessary to consider the tasks that are on the opposite side of the precedence diagram since they realistically cannot be performed in the same *station* or neighboring ones. This is done to reduce the complexity of the problem while preserving its realistic settings (Sternatz 2015). This assumption avoids considering problems that might be difficult to solve in a reasonable amount of time. The JALBFP is then solved for each section of the assembly system to find the optimal structure of the assembly line, i.e. the optimal number of *stations* and the optimal line feeding mode for the components. After, the sections of the problem are combined to obtain the final structure of the assembly line. Usually, an assembly line is not built for all its length in a straight line, but rather in segments that are sections of the assembly line with a supermarket located in the proximity where the segment's components are stored. The distance between the warehouse and the supermarket (D^{Fv}), the supermarket and the BoL (D^{vB}), and the warehouse and the BoL (D^{FB}) are 880, 750, and 1700 meters. The *cycle time* C is set to 320 seconds and the total annual production volume V is 20 000 units. The precedence diagram between the tasks and the average duration of the tasks are depicted in Figure 4.3, i.e. the precedences between the tasks.

Figure 4.4 and 4.5 depict the assignment of the tasks to the *workplaces* of the *stations* and the assignment of the components to the three line feeding modes. The implementation of the JALBFP rather than the sequential approach leads to savings 36.87% in costs. This percentage is in line with the savings that are achieved in Section 4.4.4.

Figure 4.4 and 4.5 can provide a clear explanation for the difference in the costs allocated by the solutions of the JALBFP and the sequential approach. All the tasks are assigned to the *stations* while following the precedences and time constraints in the JALBFP and the sequential approach. We can see that while the JALBFP leads to implementing one *workplace* for the majority of the *stations*, it is common for the sequential approach to implement multiple *workplaces* in the *stations*. The sequential approach leads to a shorter assembly line than the one obtained from the

implementation of the JALBFP just as in this case where the 3 fewer *stations* are implemented. From the ALFP point of view, we can see that more components are assigned to kitting and sequencing in the sequential approach rather than line stocking which is the line feeding mode that is most commonly implemented in the JALBFP. The lower number of *stations* of the solution of the sequential approach leads to the components being delivered with line feeding modes that require less space at the BoL. The sequential implementation involves solving at first the ALBP and then the ALFP. At first, the ALBP minimises the number of *stations* as can be seen in Figure 4.4 and 4.5 where the sequential approach leads to a lower number of *stations* compared to the JALBFP. After the ALBP, the ALFP optimises the delivery of the components by assigning a line feeding mode to each component. Three line feeding modes can be assigned. Kitting and sequencing require longer preparation activities and occupy little space at the BoL while line stocking needs less effort for the preparation activities but more space for storage at the BoL. After the ALBP is solved and the number of *stations* is identified, it is not possible to increase the number of *stations* so that all the components can be delivered with the cheapest line feeding mode that requires more space at the BoL. As a result, a high number of components must be delivered with more expensive line feeding modes that require little space at the BoL such as kitting and sequencing so that all the components can be placed at the BoL. For this reason, there is a large difference in the number of components that are assigned to the three line feeding modes in Figure 4.4 and 4.5. This also explains why the sequential approach leads to higher costs for the preparation activities performed at the supermarket and for the delivery of the components to the *stations* compared to the JALBFP (later shown and discussed in Figure 4.7 and Figure 4.8). The reason that makes sequencing and kitting more expensive than line stocking is that these two line feeding modes require more preparation activities. On the other hand, the JALBFP optimises in a combined way the ALBP and the ALFP. The model implements a higher number of *stations* so that there is enough space at the BoL for all the components to be delivered with the cheapest line feeding mode, i.e. line stocking. For this reason, the JALBFP leads to a higher number of *stations* as it can be seen in Figure 4.4 and 4.5. With a higher number of stations, there is enough space so that the majority of the components can be assigned to line stocking rather than sequencing and kitting. For this reason, the JALBFP assigns a higher number of components to line stocking rather than kitting and sequencing.

4.4.4 Results

The computations are performed on an 8-core server (Intel Xeon(R) Platinum 8160 CPU 2.10GHz) with 45 GB of RAM. The MILP model and ALNS heuristic are realised using Xpress-IVE version 8.5. The data generation part is performed using R (coded with RStudio of the Anaconda distribution).

Comparison with Sternatz (2015)

We compare the results of the MILP model provided in Section 4.2 with that of Sternatz (2015). Since Sternatz (2015) does not provide the data used for the experiments, we assume that ΔL_1 and ΔL_3, having the same value, do not contribute to the objective function, and $link_{i,j}$ is neglected. In Sternatz (2015), the optimisation model considers the implementation of homogeneous containers, i.e. what we refer to as line stocking, and mixed containers, i.e. kitting, that can be implemented in each assembly *station* and does not consider sequencing. Table 4.3 provides the computation times, total cost reduction, and total working time increase obtained through the implementation of our model rather than the one of Sternatz (2015).

Table 4.3: Comparison between our MILP model and that of Sternatz (2015) for 56 instances with 20 tasks J'.

	Computation times [s]		Total cost reduction [%]	Total working time increase [%]
	Our model	Sternatz		
Maximum	6988.52	2175.72	86.47	-1.14
3rd quartile	3901.7	322.11	84.86	-9.19
Median	1688.35	81.89	84.33	-14.28
Average	2374.69	310.82	83.85	-18.02
1st quartile	780.28	57.81	83.24	-27.3
Minimum	288.39	19.13	77.84	-48.62

While our model minimises the total costs of the whole assembly system (as explained in Section 4.2), Sternatz's model minimises the total working time of the whole manufacturing system as the sum of the assembly time activities and the part feeding activities to produce one product unit ($\frac{hours}{product}$). The total working time is the sum of

the time for the part feeding operations and the cycle time multiplied by the number of *workplaces*. This time can be considered the labour costs of some operations that are performed in an assembly system. Our model has higher computations times and always leads to a cost reduction compared to the solution from the model proposed by Sternatz (2015). This total cost reduction is achieved by an optimal mix of the feeding policies which considers not only kitting but also sequencing. For the total working time, neither model consistently leads to an increase or reduction. However, the total working time increases for some instances with respect to Sternaz but the total costs for the whole system always decrease.

Validation of the Heuristic

Before we validate the heuristic, we tune its parameters to ensure that it is effective. In other words, we select the minimum number of iterations η_1 and the number of iterations performed after the last feasible improved solution is found η_1' to ensure that the algorithm is effective when solving the JALBFP. Solving the JALBFP requires more time than solving the sequential approach. Solving the sequential approach consists in solving the ALBP and the ALFP. Both these problems are a specific version, subproblem, of the JALBFP with the relaxation of some constraints and without some elements in the objective function. For this reason, we tune the parameters, η_1 and η_1', to solve the JALBFP and we are going to apply these parameters also to solve the sequential approach. One might argue that the values for η_1 and η_1' should be different when solving the JALBFP and the sequential approach. However, the JALBFP is more difficult to solve than the sequential implementation of the ALBP and the ALFP. For this reason, we select the values for these two parameters for the problem that is more difficult to solve and apply them also to the other one. Although this might lead to higher computation times for the sequential approach, it cannot prevent the sequential approach from finding the best solution. In other words, since the heuristic is effective when solving the JALBFP with these two parameters, it will be also effective in solving the sequential approach. We tune the parameters of the ALNS heuristic by comparing it to the MILP model detailed in Section 4.2. We run the MILP and the heuristic to solve 42 'less tricky' and 27 'tricky' instances of 20 tasks J'. Figure 4.6 provides the *computation times* before the heuristic stops and the deviation between the results obtained from the optimisation model and the heuristic.

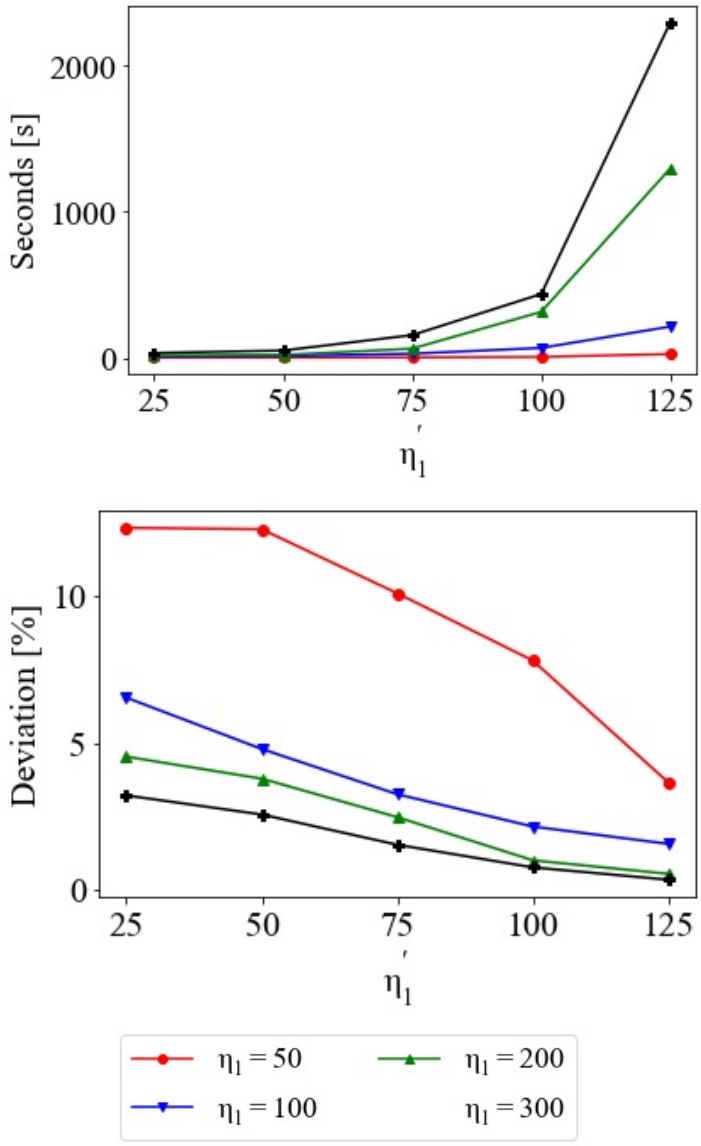


Figure 4.6: Average computation times and cost deviation with the tuning of the parameters η_1 and η_1' of the ALNS heuristic.

With a low value of η_1, 50 or 100, the algorithm is not effective in decreasing the deviation since on average it might even reach values above 10%. This is because during the first 50 or 100 iterations, the algorithm is not able to find an improved solution and it stops prematurely. On the other hand, when η_1 is equal to 200 or 300, the deviation decreases significantly. However, we need also to select the right η_1', and this might lead to having high *computation times*. When η_1' is equal to 25 or 50, the effectiveness of the algorithm in the quality of the solution, the deviation, decreases.

This is because the algorithm requires more than 25 or 50 iterations to find a better solution after the last one is found. A combination of a high value of η_1, e.g. 300, and η_1', e.g. 125, leads to the best results for the deviation but prohibitive *computation times*. For this reason, we select η_1 equal to 200 and η_1' to 100. With this value, the average deviation is around 1% and the average *computation time* is lower than the maximum and around 350 seconds.

We validate the ALNS heuristic by comparing it to the MILP model detailed in Section 4.2. For the JALBFP and the sequential approach, we compare the MILP model and the heuristic explained in Sections 4.2 and 4.3. For the sequential approach, the objective function considers only the costs of the ALBP activities ($C_I^i + \omega C^a$). We run the MILP model with a time limit of 7 200 seconds for the instances from the set of Otto, Otto, and Scholl (2013) and selected 75 'less tricky' and 72 'tricky' instances that contain 20 tasks J'. To reduce computation times, the value for Big M in constraints (4.11, 4.13, and 4.14) is set as low as possible but large enough to avoid problems with the optimisation model, e.g. just a little higher than the cycle time C. The MILP model is not able to solve instances with 50 or more tasks from the same data set. To test the joint model, we vary the $\frac{Total\ volume\ of\ components}{Total\ task\ time}$ and the $\frac{Number\ of\ components}{Total\ task\ time}$ between 0.0005 and 0.0020 $\frac{m^3}{s}$ and between 0.015 and 0.030 $\frac{1}{s}$ and variable distances between the warehouse, supermarket, and assembly line (D^{FB}, D^{Fv}, and D^{vB}). In Table 4.4, we provide the deviations between the MILP model and the heuristic. The maximum number of assembly *stations* S' is set to 15 *stations*, but the solution of all instances for the JALBFP and the sequential approach implements no more than 10 *stations*. For the ALNS algorithm, η (the minimum number of iterations) and η_1 (the number of iterations performed after the last improved solution is found) are empirically set to 200 and 100 to avoid excessive computation times and to ensure that the algorithm renders good solutions. We replicate the heuristic 30 times for each instance in order to ensure that the results are consistent. For each measure, we provide the results obtained.

Table 4.4: Deviation between optimisation model and the heuristic for 147 instances with 20 tasks J'.

	Deviation [%]	
	JALBFP	Sequential approach
Maximum	10.75	7.89
3rd quartile	0.78	0.0
Median	0.23	0.0
Average	1.0	0.95
1st quartile	0.0	0.0
Minimum	0.0	0.0

The deviation between the optimal solution and the ALNS algorithm has a maximum of 13.52%, but the median and the average of about 1%. This shows that the ALNS algorithm is effective at solving the JALBFP and the sequential approach in a large majority of the cases, but there are few instances where the ALNS algorithm yields a higher deviation. The results show that the computation times for the heuristic are lower than those of the MILP model for both the average and the median. The minimum computation time is higher than those of the MILP model, but it yields an average time for both the JALBFP and the sequential approach. This is because the ALNS algorithm requires at least a certain amount of time to be effective, but it takes more time with instances that are difficult to solve.

Comparison Between the JALBFP and the Sequential Approach

To show the benefits of implementing the JALBFP and the sequential approach, we solve 54 large 'less tricky' instances and 18 large 'tricky' instances with 100 tasks J' and 102 medium 'less tricky' and 39 medium 'tricky' instances with 50 tasks J' from the set of Otto, Otto, and Scholl (2013) with the ALNS algorithm described in Section 4.3. The heuristic is not able to solve any very large-scale instances (with 1 000 tasks J) in a reasonable amount of time. One could argue that a heuristic should be able to solve instances with few hundreds tasks. However, there are two issues to consider in this argument.

First, our heuristic relies on a MILP for the ALFP. This is because to this day no approach employs a metaheuristic or heuristic to solve the ALFP. Therefore, in an

instance with few hundreds tasks there would be thousands of components that need to be assigned a line feeding mode. The line feeding mode selection for these components relies on a MILP with long computation times.

Second, solving instances with few hundreds tasks is not possible in a reasonable amount of time. Our algorithm solves instances that are composed of 20, 50, and 100 tasks. Of course, solving an instance of 100 tasks needs more time than an instance of 20 tasks because 100 tasks could be arranged differently in the stations, i.e. there are more combinations for the assignment of the tasks to the stations with 100 tasks rather than 20 tasks. For this reason, the stopping criterion of the ALNS is to perform at least η_1 iterations and to continue for η_1' iterations after the last feasible improved solution is found. With few hundreds tasks, the computation time increases because there is a higher number of combinations of the assignment of the tasks to the stations and the algorithm could require more time to find a better solution compared to when there are only 20 tasks. For this reason the computation time for a very large-scale instance with few hundreds tasks becomes prohibitive.

In the sequential approach, we first solve the ALBP without any ALFP constraints (4.15-4.21) and with an objective function that considers only the assembly costs ($C_I^i + \omega C^a$). Then, we solve the ALFP with the solution obtained from the ALBP, i.e. the assignment of the tasks to the *workplaces* and *stations*. Since the solution from the ALBP could be infeasible with the introduction of the ALFP constraints, i.e. there is not enough space at the BoL to store the components, we introduce the parameter ψ which represents the possibility to perform multiple deliveries to the BoL. The parameter ψ divides the space, weight, and volume of the components that are delivered in constraints (4.16), (4.18), and (4.19). The parameter ψ is also introduced as a multiplier in the objective function (4.4.8), (4.5.2), (4.5.3), and (4.5.4) of the ALFP to account for the costs of the multiple deliveries. The ratio $\frac{Number\ of\ components}{Total\ task\ time}$ is equal to the one found in a manufacturing environment. The maximum number of assembly *stations* S' is set to 50, but the solution of all instances for the JALBFP and the sequential approach implements no more than 30 *stations*.

Table 4.5 provides the *worker densities*, the *component densities*, *idle times*, *computation times*, and *total cost reductions* for the solutions obtained from the JALBFP and the sequential approach, i.e. the sequential implementation of the ALBP and the ALFP.

Table 4.5: *Worker densities*, *component densities*, *idle times*, *computation times*, and *total cost reductions* of the heuristic for 54 large 'less tricky' instances and 18 large 'tricky' instances with 100 tasks J' and 102 medium 'less tricky' instances and 39 medium 'tricky' instances with 50 tasks J'.

	Worker density		Component density		Idle time [%]	
	JALBFP	Sequential approach	JALBFP	Sequential approach	JALBFP	Sequential approach
Maximum	1.71	2.62	34.0	55.0	0.63	0.63
3rd quartile	1.37	2.0	26.15	37.78	0.57	0.57
Median	1.18	1.75	24.38	33.75	0.55	0.55
Average	1.23	1.67	24.49	32.9	0.54	0.55
1st quartile	1.08	1.3	22.73	27.27	0.51	0.53
Minimum	1.0	1.0	16.92	17.37	0.43	0.47

	Computation time [s]		Total cost reduction [%]
	JALBFP	Sequential approach	
Maximum	96650.16	13683.9	43.82
3rd quartile	12109.02	2231.81	11.97
Median	5894.74	1434.67	8.33
Average	11095.87	2237.44	10.07
1st quartile	3066.24	770.88	5.08
Minimum	953.0	22.85	0.32

The results show that the sequential approach leads to *worker densities* that are higher than the ones in the JALBFP. The reason for this is that the sequential approach is performed by implementing the ALBP as a first step, which aims at decreasing the costs of the assembly *stations*. This leads to fewer *stations*.
Similarly, the *component densities* are lower for the JALBFP than those with the sequential model.
We also analyse the *idle times*, i.e. the percentage of *cycle time* spent in non-productive activities. In this case, there is no clear difference between the two approaches. This shows that minimising the *idle time* to solve the JALBFP does not guarantee that the solution would be optimal.
The *computation times* of the JALBFP are higher than the ones of the sequential approach. Although in some cases the JALBFP has long computation times, this is a problem that is solved at a tactical level. The average computation time is lower than 1 hour, which is reasonable for a problem that is solved at the tactical level.

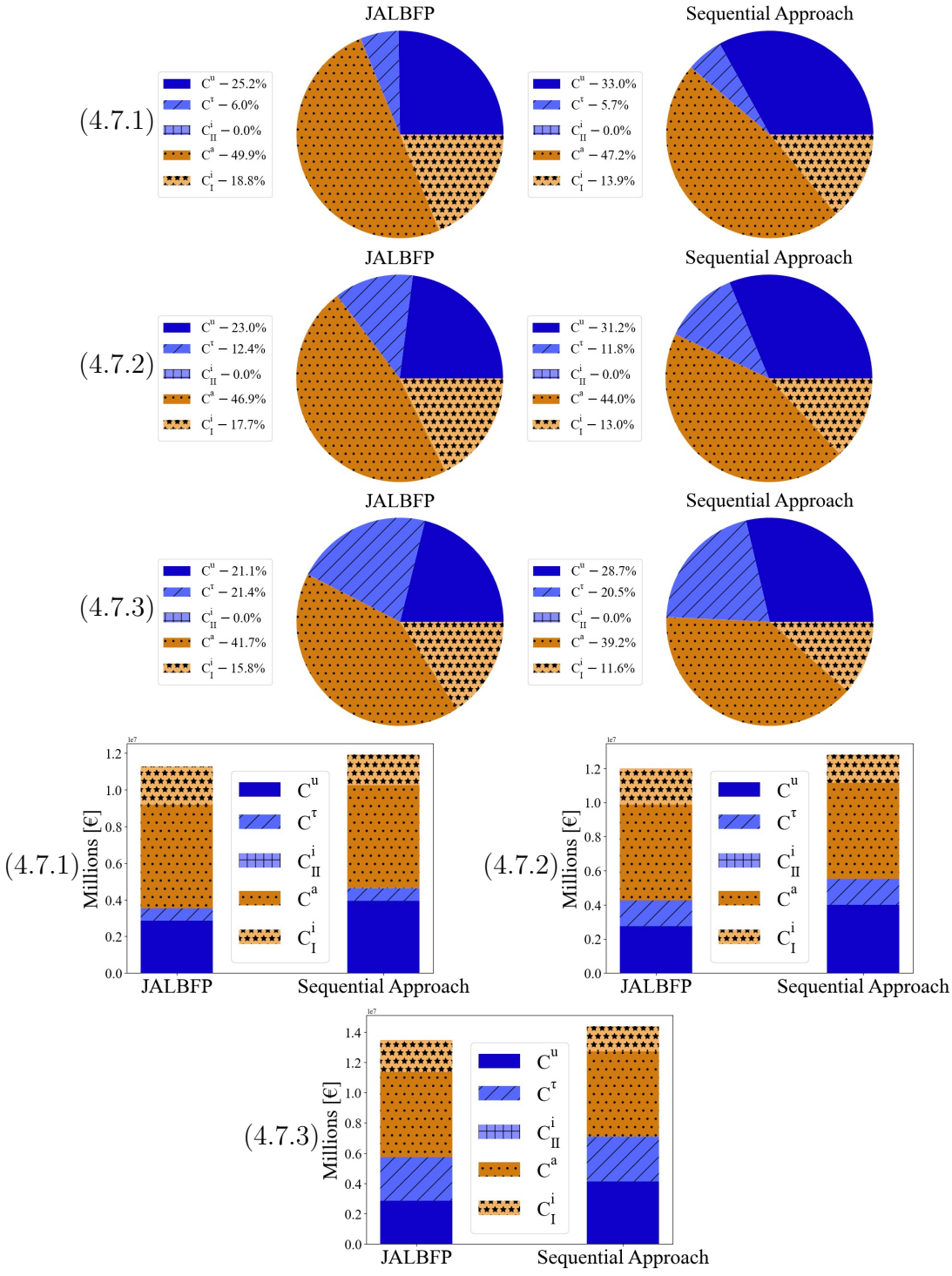


Figure 4.7: Costs of the JALBFP and the sequential approach for one instance with 100 tasks J' and variable distances (D^{FB}, D^{Fv}, and D^{vB}). The costs in blue refer to the ALFP activities while the costs in orange refer to the ALBP activities.

The most relevant result is that solving the JALBFP leads to a total cost reduction compared to the sequential approach. The average total cost reduction is 17.52%. The ALFP affects the operating costs of an assembly line, and in some cases, the total cost reduction reaches 56.53%.

We analyse the causes of the total cost reduction for one instance with variable distances (D^{FB}, D^{Fv}, and D^{vB}). In Figures 4.7 and 4.8, we compare the solutions found by the heuristic for both the JALBFP and the sequential approach.

In Figure 4.7, the pie charts show the percentages of the costs. The sequential approach consistently leads to supermarket costs that are higher than those for the JALBFP. At the same time, the transportation costs are expectedly higher with the maximum distances between the assembly line and the warehouse and the supermarket (4.7 - 4.8.3). Figure 4.7 also shows the stacked charts for the costs of the JALBFP and the sequential approach of one instance with 100 tasks J'. These are costs that are incurred over a period of 5 years of operating an assembly line. The costs for the JALBFP are lower than those for the sequential approach with all distances. Here, the transportation costs (C^{τ}) and supermarket costs (C^{u}) are consistently higher for the sequential approach than for the JALBFP. Both costs increase with the maximum distances between the warehouse and assembly line (4.7 - 4.8.3). In contrast, the investment costs for the assembly line increase in the JALBFP. In Figure 4.7, we can see that the supermarket and transportation costs are higher for the sequential approach than for the JALBFP.

Figure 4.8 provides the details of the assembly line with variable distances (D^{FB}, D^{Fv}, and D^{vB}). It can be observed that there is consistently a higher number of *stations* in the JALBFP. In contrast, there is a higher number of components that are kitted or sequenced in the sequential approach. The lower number of *stations* for the sequential approach explains why the investment costs for the assembly line activities are lower. This is because the sequential approach consists of the sequential implementation of the ALBP and the ALFP. To begin with, the ALBP minimises the number of *stations*. Then, the ALFP must assign certain components to the kit and sequencing line feeding mode because of the low space and the low number of *stations*. In other words, the aim of first minimising the number of *stations* leads to higher preparation and transportation costs.

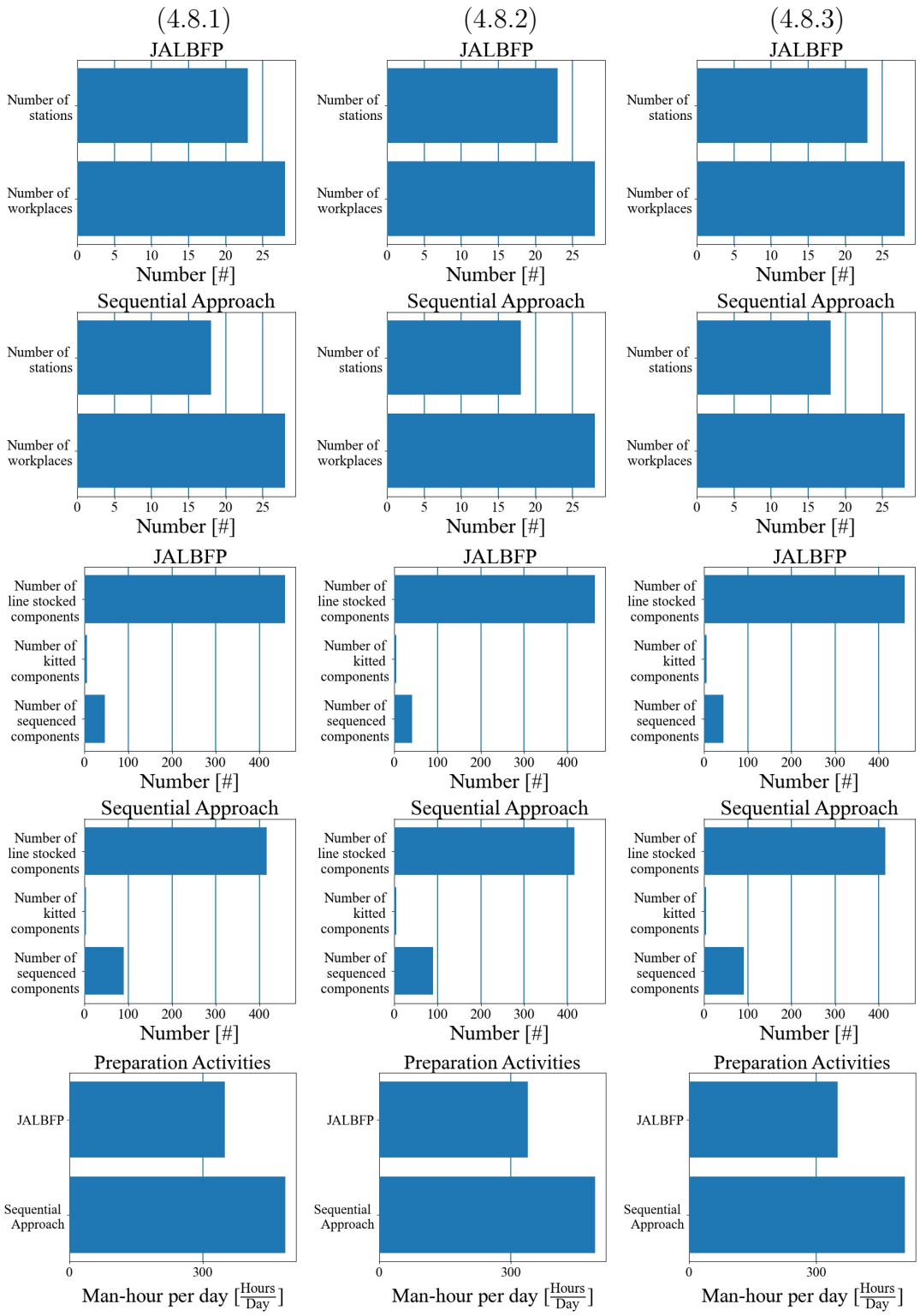


Figure 4.8: Details of the assembly line for the JALBFP and the sequential approach for one instance with 100 tasks J' and variable distances (D^{FB}, D^{Fv}, and D^{vB}).

For the preparation activities that are performed to deliver the components with the line feeding modes, we can see that the sequential approach consistently leads to a higher value of man-hour per day. This means that a higher number of operators must perform these activities in the sequential approach rather than the JALBFP.

Sensitivity Analysis

We perform a sensitivity analysis to better understand in which situation solving the JALBFP is economically more convenient than using the sequential approach.
We solve the large 'less tricky' instances from the set of Otto, Otto, and Scholl (2013) that contain 100 tasks J' using the ALNS algorithm described in Section 4.3. The number of components P', the volume of components delivered to the *stations*, and the length at the BoL L^s, i.e. where the components are stored, vary to show how they affect the total cost reduction. We include some extreme cases where there is a high number and a high volume of components that are delivered to the assembly stations. Figure 4.9 depicts the total cost reduction that can be achieved when these measures vary.
We use linear regression to show the relationship between the measures and the total cost reduction. The total cost reduction decreases with the number of components delivered to the *stations*. Similarly, the total cost reduction decreases with the total volume of components delivered to the *stations*. When there is a lower number of components or components with a lower volume, these components can all be line stocked. For this reason, the total cost reduction between the JALBFP and the sequential approach decreases with a lower volume of components. In contrast, a different behaviour can be seen when the length at the BoL increases. As we increase this length, there is more space to store the line stocked components at the BoL of the *stations*. For this reason, the total cost reduction decreases with the increase of the length at BoL. In other words, a longer BoL also means the possibility to line stock the components.

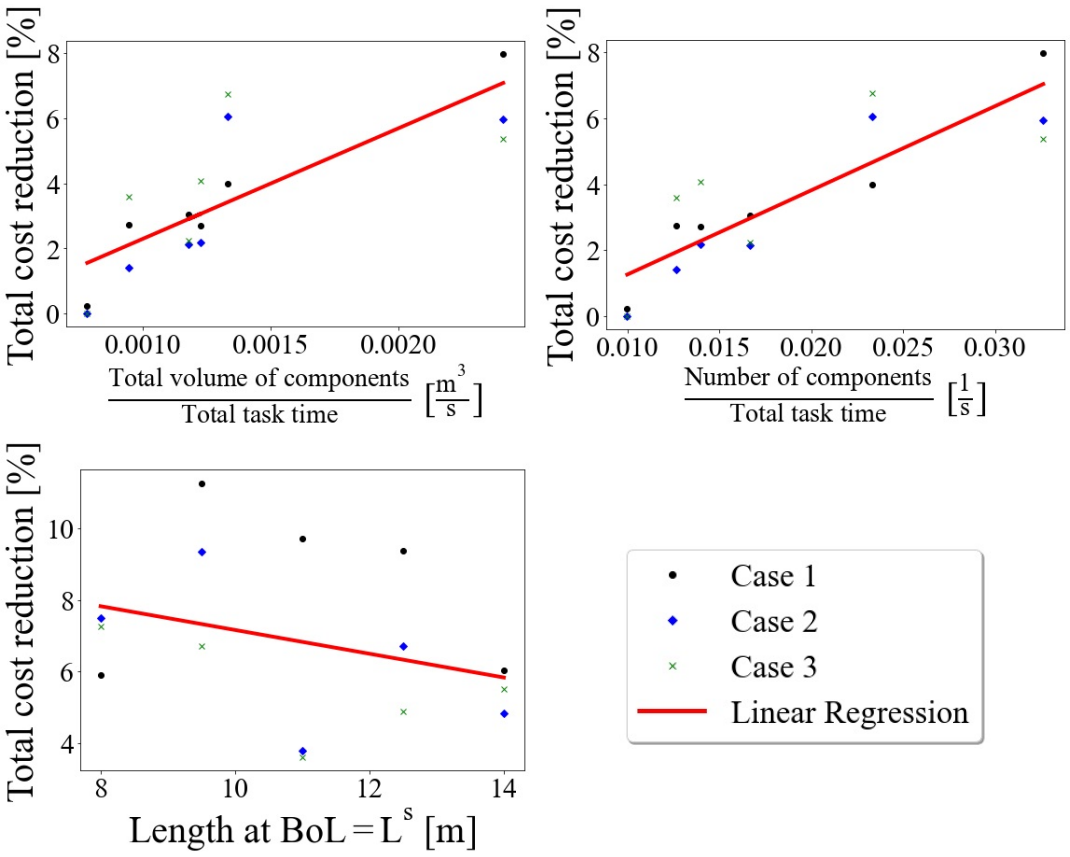


Figure 4.9: Total cost reduction of the JALBFP vs. the sequential approach for 15 instances with 100 tasks J' with varying length at BoL (L^s), $\frac{Total\ volume\ of\ components}{Total\ task\ time}$, and $\frac{Number\ of\ components}{Total\ task\ time}$.

4.5 Conclusion

Herein, a MILP model and an ALNS are proposed to solve the multi-manned JALBFP by considering different task times for each product model, multiple *workplaces* per *station*, the kitting and sequencing line feeding mode, and a cost objective function that considers all the operations performed in an assembly system. The major contribution of this model consists of the optimisation model and the heuristic that have a high level of detail compared to previous models in the literature. Unlike previous models that consider the ALFP and the ALBP in a sequential approach, the proposed algorithms solve them in a combined way. We compare our MILP model to that of Sternatz (2015) and find that it always leads to lower total costs. We validate the ALNS

algorithm and we find that the average deviation is around 1%. Our approach can be used for balancing operations of a new assembly line or for rebalancing operations of an existing assembly line. We show that there is an average and maximum total cost reduction of 17.52% and 56.53% if the JALBFP, rather than the sequential approach, is implemented. In the sensitivity analysis, we show the condition when the total cost reduction between the JALBFP and the sequential approach decreases. We also provide an example of the application of the model to a real-world problem of a company that assembles mini-buses. The ALNS algorithm is already able to solve large-scale instances with 100 tasks and 'tricky' instances with 50 tasks and this is a step ahead of what is done in Sternatz (2015) and Calzavara et al. (2021). A limitation of our model is that is not able to solve instances with few hundreds tasks J' of 'tricky', 'extremely tricky', or 'open' instances. Future research could try to solve 'extremely tricky' and 'open' instances as well as instances with few hundreds tasks.
Sternatz (2015) and Battini et al. (2017) provide a model for the JALBFP. These articles consider the direct and indirect line feeding mode. The direct line feeding mode consists in deliverying components directly from the warehouse to the assembly line. The indirect line feeling mode consists in deliverying the components at first to a warehouse where preparation activities are performed before the components are delivered to the assembly line. In other words, the indirect line rewind mode encompasses the sequencing and the (traveling) kit. Further, these articles minimise the number of assembly and logistic operators and the time to perform the operations. The model and algorithm provided in this chapter consider the JALBFP with a higher level of detail. We consider three line feeding modes, namely line stocking, sequencing, and traveling kit. We also minimise the costs of the assembly and logistic operations. Thanks to the higher level of detail, we can understand the actual implications of solving the JALBFP rather than the sequential approach. With these approaches, we can understand the amount of savings that can be generated from this switch.
The algorithm proposed in this chapter shows one limitation that can be better investigated in future research. The approach provided is affected by the size of the instances solved. This is because, we rely on the MILP model for the line feeding mode selection. Further research should investigate new methods to solve the JALBFP. This can be done by applying existing methodologies, adjusting existing algorithms, or developing new ones. Future research should provide an algorithm to solve instances of the JALBFP regardless of their size. Further research should investigate the performance of the model and the algorithm and the savings of JALBFP model when

solving instances with different characteristics (order strength and peak, two measures that encode the shape of the precedence diagram). A sensitivity analysis on this subject could help explain under which circumstances the JALBFP can be more easily solved or is more economically convenient. Future research should also investigate how a very large-scale instance (more than 200 tasks) can be divided into multiple smaller instances. Not only the number of tasks, but also the kind of tasks and the number of components could play a role in the results of the model and algorithm. This is necessary to understand how these smaller instances should be conceived for the ALBP and JALBFP.

Chapter 5

Routing and Scheduling Activities for the Delivery of the Components

Based on

Zangaro F., Battini D., 2018. "Integrating Routing and Scheduling Decisions for Enhancing the Part Feeding in an Automotive Environment", *Twentieth International Working Seminar on Production Economics*

In an automotive environment, part-feeding procedures involve the delivering references that are stored in a supermarket to the stations. The part-feeding procedures involve a high number of tow trains that travel through a limited network of roads. This leads to congestion problems that can disrupt the delivery of references. To solve these congestion problems, we define the Inventory Routing and Scheduling Problem (IRSP). A Mixed Integer Linear Programming (MILP) model is provided for the IRSP in order to integrate the tow-train routing into the schedules of the delivery, which includes an arrival, *serving* or waiting, and departure time for each station. This approach minimises the delivery time to the stations by considering also congestion problems. For real-life implementation, a heuristic that is based on a Large Neighbourhood Search (LNS) is provided. We find that the deviation between the objective value of the optimisation model and the heuristic is lower than 2%. A numerical case is outlined to discuss the benefits and performance. We find that congestion problems can lead to an increase in the delivery time of 16.5%. Based on the literature, we consider the congestion problem that occur in three different layouts that are common in a manufacturing environment: product line layout, fixed product layout, and process layout. We study in which layout congestion problems occur more commonly and we find that the process layout causes the largest amount of congestion problems.

5.1 Introduction

The part-feeding process is a complex problem because it concerns many elements and factors. This process is important in the automotive industry and its satellite environments. It involves the delivery of references from the supermarket to the assembly stations. A great number of resources and operators are involved in these processes. Therefore, automotive companies strive for improvements in order to achieve higher productivity and lower costs.

It has been shown that congestion and traffic problems can affect the logistic practices of many transportation environments (Johnson 2001; Roy, Gupta, and De Koster 2015). Consequently, congestion issues can also affect a wide range of operations such as the part-feeding process.

The part-feeding process involves the transport of components, also known as references, from the supermarket, where they are stored, to the assembly stations, where they are needed for the assembly operations, see Figure 5.1. These delivery operations are of great importance for the assembly or production activities carried out at the stations. Tow trains, which are vehicles composed of an engine and a few wagons, are used for such delivery practices. In a production plant, roads are of limited size and scarce (Battini et al. 2015). Due to high traffic and safety issues, overtaking operations are often forbidden. Therefore, when a train is loaded or unloaded at a station, other arriving trains must wait in line. A large amount of time might be lost due to queues. Such congestion problems might lead to low utilisation rates and stock-out situations. To account for these problems, a higher number of tow trains might be used for performing these operations, which then causes more queues. We define this problem as the periodic Inventory Routing and Scheduling Problem (IRSP) that aims at routing and scheduling the delivery of references to the assembly stations by considering the congestion problems and the queue time. In order to minimise the time spent on delivering activities, the IRSP considers the traveling time of the vehicles and the consumption rate of the stations. This ensures that each station receives the references in time so that no stockout situation occurs. At the same time, congestion problems for the train drivers are avoided. Figure 5.1 provides a map of a production plant and an example of the delivery schedules of the tow train that are periodically performed.

In this chapter, we provide a mathematical formulation and a heuristic to solve the periodic IRSP. This formulation considers congestion situations that can occur during

the delivery of the components to the production or assembly area. The formulation reduces congestion problems so that the least amount of time is spent by operators in queues and the delivery of components. Through this formulation, congestion problems in the delivery of components can be reduced so that operators do not waste time in such unproductive activities on the assembly line. A clear schedule is provided to the drivers of the tow trains and, when possible, congestion problems are avoided. This schedule might include a waiting time at the supermarket before delivering the components. During this time, the drivers can perform some picking activities in the supermarket if the schedule allows it. The formulation schedules the periodic delivery of the minimum amount of components to stations so that the production activities can be performed. To show that the heuristic is effective, its results are validated with the ones of the optimisation model, and we find that the average and median deviation is lower than 2%. We find that solving the IRSP rather than the IRP leads to an average time deviation, i.e. a reduction of the roundtrip time, of 16.5%. We also perform a study with different layouts (Tompkins et al. 2010) to identify those layouts where the congestion problems are more frequent and where the implementation of the IRSP is most beneficial. In the sensitivity analysis, we implement the IRSP for three common layouts that can be found in a manufacturing environment, product line layout, fixed product layout, and process layout. This analysis shows which layout is more likely to yield higher savings in terms of time. This explains when it is more convenient to solve the IRSP rather than the Inventory Routing Problem (IRP). This could also justify, when possible, a change of layout in favour of one that causes less congestion problems.
The rest of this chapter is organized as follows: Section 5.2 describes the part-feeding problem, Section 5.3 describes the mathematical formulation of the mixed integer linear optimisation approach and the heuristic, Section 5.4 validates and implements the heuristic, and Section 5.5 provides a summary to the chapter.

5.2 Problem Description

The supermarket concept is commonly implemented in the automotive industry (Battini, Boysen, and Emde 2013). References, i.e. components, are stored in cardboard boxes or plastic containers of different dimensions. Three line feeding modes are implemented for the delivery of the components: line stocking, kitting, and

sequencing. Line stocking consists of the delivery in a single box of multiple identical parts of the same component.

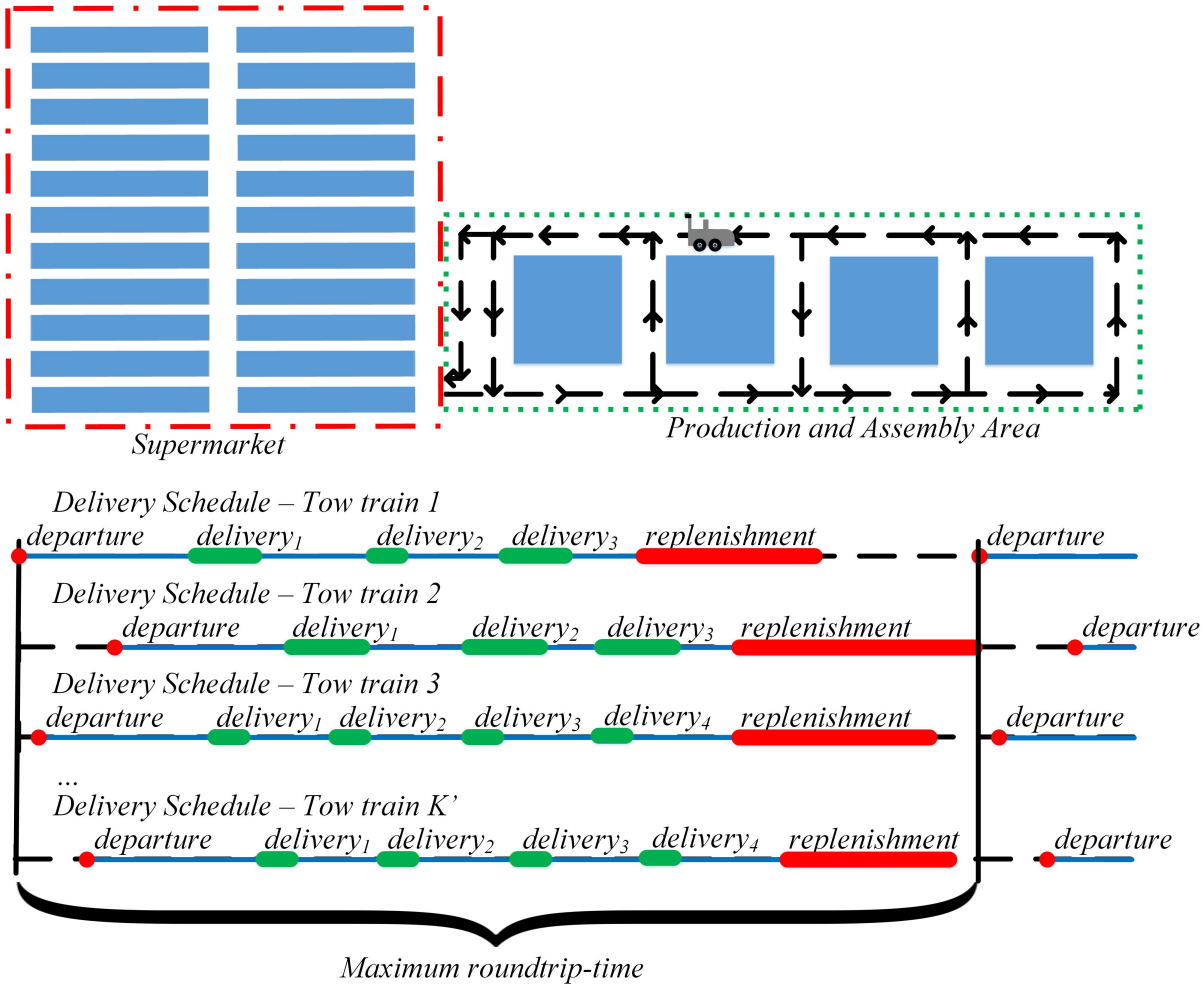


Figure 5.1: Map of a production plant and delivery schedule of the tow trains.

Kitting is a container with different components that must be delivered at the same station. Sequencing consists of delivering different variants of the same component in the same container in the same order as they are needed in the assembly operations. Picking activities, which are necessary for kitting and sequencing, consist of collecting the references from their boxes so that they can be placed in the right container. These picking activities are the most human-intensive operations performed in a warehouse since the operators must travel to different locations of the supermarket to collect all

the components. The picking activities performed in a warehouse account for more than 50% of its expenses (Richards 2014; Setayesh et al. 2022; Tompkins et al. 2010). The picking activities are performed by logistic operators that work at the supermarket but they can also be supported by additional workers. To decrease the costs of the picking activities, idle operators could support these operations. Potential savings can be achieved if the idle operators can spend some time performing these activities. Though, the idle operators must do this promptly to return to their primary operations if required. For delivering the components, operators use tow trains (or tugger trains), which are vehicles usually composed of an electrically powered engine and a few wagons that perform the delivery procedures (Battini et al. 2015). All the tow trains available in a plant are generally utilised for the part-feeding process. This is because there is a need to balance the workload among all the available tow trains. Firstly, the operators unload at the warehouse all empty containers or boxes that they had previously collected from the stations. These containers are then disposed of or returned to the supplier. The operators load the full containers and boxes on the tow trains before departing the supermarket. As soon as all the containers are loaded, the tow trains can proceed to deliver them to the assembly stations based on the schedule assigned to them. We assume that the departure order of the tow trains does not affect the arrival at the stations. To reach different locations of the production plant, tow trains use a few small roads. These roads are limited in dimension and number to use as much space as possible for productive or assembly operations in a plant, and the network of roads might, in some cases and based on the layout of the production plant, be quite intricate. While a tow train drives towards its destination, it passes other stations. Because of this, it is important to notice that there is a difference between *serving* and *visiting* a station. A train can *visit* many different stations when it is traveling in the production area. In this case, no reference is delivered to these stations, but still a train arrives and departs. On the other hand, a station is *served* when a train delivers some references. If a tugger train *serves* a station, it must stop to perform delivery operations. We assume that all the stations have a consumption rate higher than 0, and, thus, must be *served*. At the same time, we assume that each station must be *served* by one vehicle only. We assume that the delivery time is independent of the kind of container used, i.e. a box, a sequencing container, or a kitting container. We also assume that the delivery time of the containers include the delivery of the full containers from the tow train and the retrieval of the empty ones from the racks at the stations. In such operations, the filled containers are unloaded from the tow trains, and

the empty containers are loaded onto them. Due to the limited size of the roads, the restricted space, and for reasons of safety, overtaking operations are forbidden to tow trains. Therefore, any tow train that encounters another one while performing delivery procedures must wait in queues. We assume that a train that is *visiting* a station must wait in queue if it arrives anytime while that station is being *served*. This occurs even if the *visiting* train arrives at the same time as the *serving* train.

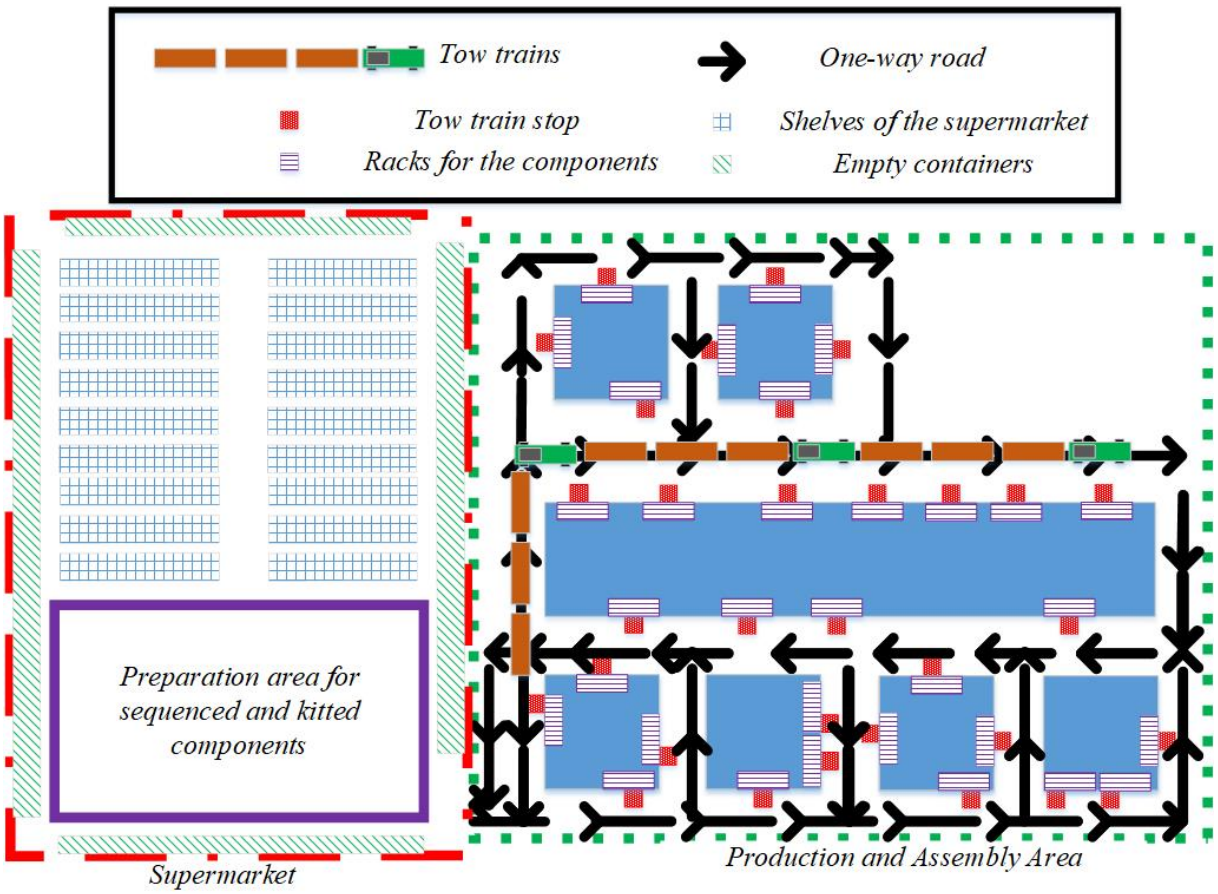


Figure 5.2: Part feeding activities performed in an assembly or production plant.

This situation can occur more than once in a tow train's delivery trip and it might also occur in the replenishment process at the supermarket. If these congestion problems are avoided, a larger amount of time can be spent by operators in the supermarket while performing picking activities. Through the routing and the scheduling of tow trains, we can avoid two tow trains meeting while one of them is performing the delivery of

components. In the most extreme scenarios, one tow train can wait at the supermarket and help with the picking activities to avoid congestion problems. On the other hand, if congestion problems cannot be avoided, they can be at least minimised. After a tow train performed all its delivery operations to all the stations that were assigned to it, it returns to the supermarket for the refilling operations. If there is some spare time between two consecutive roundtrips, the driver helps the logistic operators in the supermarket to perform the picking activities. If a train is delayed in its delivery trip, it must deliver, in the next roundtrip, a larger amount of references to the stations to replenish the depleted safety stock during its next roundtrip. This side effect leads to an unstable and unreliable length of roundtrip.

5.3 Mathematical Model

This section provides an explanation of the Mixed Integer Linear Programming (MILP) model and the heuristic. Although the MILP model aims at reaching the optimal solution, it cannot solve any real-life problem in anything shorter than a prohibitive computational time. Because of this, a heuristic is provided. This approach aims at finding near-optimal solutions based on a Large Neighbourhood Search (LNS).
The result of these approaches is a delivery schedule that includes an arrival, a serving, and a departure time for each station. This schedule consists of a list of all the stations that are *served* and *visited* by each train. If a train *serves* a station, there is an arrival time, a delivery time, and a departure time. On the other hand, if a train *visits* a station, an arrival time, a queue time, and a departure time are provided. Since the approaches consider also the stations that are *visited*, the tow trains can travel only through adjacent stations. i.e. nodes of the network that are next to each other. The delivery schedules are repeated continuously for as long as the stations are supplied with references.

5.3.1 Mixed Integer Programming Model

This model is a three-index formulation that solves the IRSP. Although other two-index formulations have also been implemented for VRP, it is not possible to develop a two-index model for the IRSP because it is necessary to understand which train *serves* and *visits* which location. A two-index formulation only allows the development of routes for the trains, but it neglects how these routes are assigned to the trains. There

are N stations *served* by a single supermarket. Each station is *served* by one of the K trains. c_{nm} is the distance between station n and station m. The demand rate d_n is measured in cubic meters per minute for each station n, and the demand rate f_n in boxes per minute. All the racks of each station n have a maximum load of L_n in cubic meters. Matrix a_{nm} is 1 if there is a direct road from station n to station m. The unloading and loading times of a box are equivalent and encoded in s, which is measured in minutes for each box. Each vehicle k has a maximum capacity load M_k that is measured in cubic meters.

Table 5.1: Sets, parameters, and decision variables.

Sets of the model	
Not.	Definition
k, h	Indicis of tow trains for the delivery
$K = 1, \ldots, K'$	Set of tow trains for the delivery
n, m	Indicis of stations for the delivery of the components
$N = 0, \ldots, N'$	Set of stations for the delivery of the components, $n = 0$ denotes the supermarket
Parameters of the model	
Not.	Definition
a_{nm}	Direct roads between station n and station m (binary, equal to 1 if that is the case, 0 otherwise)
c_{nm}	Travel times between stations n and m
d_n	Consumption rate of station n in volume
f_n	Consumption rate of station n in boxes
L_n	Maximum stock at station n
M_k	Maximum loading volume for train k
s	Loading and unloading time per box at a station
Decision variables	
Not.	Definition
b_{kn}	Variable that encodes if train h which *serves* station n arrives to it before than the departure of the train k (binary, equal to 1 if that is the case, 0 otherwise)

e_{kn}	Variable that encodes if train h which *serves* station n departs from it later than the arrival of the train k (binary, equal to 1 if that is the case, 0 otherwise)
q_{kn}	Departure time of train k from station n
r_{kn}	Resupply quantity at station n delivered by train k (integer)
t_k	Roundtrip time of train k
u_{kn}	Arrival time of train k at station n
v_{kn}	Loading volume when train k leaves station n
w_{kn}	Waiting time of train k at station n
x_{nmk}	Indicator if train k travels from station n to m (binary, equal to 1 if that is the case, 0 otherwise)
y_{kn}	Indicator if train k *serves* station n (binary, equal to 1 if that is the case, 0 otherwise)

Objective Function

$$\min T_{tot} = \sum_{k \in K} (t_k - q_{k0}) \tag{5.1}$$

The objective function (5.1) considers the traveling time of all the vehicles T_{tot} after the departure from the supermarket. This time is the sum of the roundtrip time t_k of all the vehicles K'. In order to calculate the time spent while delivering the components and not at the supermarket before departure, we deduct the time of the departure from the supermarket q_{k0}.

Constraints

We divide the constraints into *routing constraints*, *timing constraints*, *loading constraints*, *valid inequalities*, and *variable definition*.

Routing Constraints These constraints refer to the route taken by the tow trains in the delivery of the components to the stations.

$$\sum_{k \in K} y_{kn} = 1, \quad \forall n \in N \tag{5.2}$$

$$y_{kn} \geq \frac{r_{kn}}{M}, \quad \forall n \in N, k \in K \tag{5.3}$$

$$y_{kn} \leq \sum_{m \in N} x_{nmk}, \quad \forall n \in N, k \in K \tag{5.4}$$

$$\sum_{m \in N} x_{mnk} = \sum_{m \in N} x_{nmk}, \quad \forall k \in K, n \in N,\ m \neq n, a_{mn}, a_{nm} = 1 \tag{5.5}$$

$$\sum_{n \in N} x_{0nk} = 1, \quad \forall k \in K \tag{5.6}$$

Constraints (5.2) enforce that each station n is *served* by one train. Constraints (5.3) ensure that if a container r_{kn} is delivered to a station n by train k, then this train must *serve* it, i.e. y_{kn} equal 1. Constraints (5.4) ensure that if a station n is *served* by a train k (y_{kn} equal 1) then the train must travel to that station. The supermarket ($n = 0$) is not considered in this set of constraints. Constraints (5.5) ensure that each train k that *visits* a station n traveling from a station must depart from it to *visit* another adjacent station. This is possible only if n and m are adjacent stations, i.e. a_{mn} and a_{nm} is equal to 1. This is done for all the stations and the supermarket ($n = 0$). In a production plant, all the available vehicles are used to deliver the components to the stations. Constraints (5.6) ensure that all the vehicles K' depart from the supermarket for the delivery of the components to the stations N'.

Timing Constraints These constraints refer to the time needed by the tow trains to travel and deliver the components. Similarly to an IRP, we need to calculate the time necessary for the delivery of the component to calculate the quantity that is delivered to the stations.

$$u_{kn} \geq q_{km} + c_{mn} - \underbrace{(1 - x_{mnk})M}_{5.7.1}, \quad \forall k \in K, m, n \in N, m \neq n, a_{0n} = 1 \tag{5.7}$$

$$u_{kn} \geq c_{0n} - \underbrace{(1 - x_{0nk})M}_{5.8.1}, \quad \forall k \in K, n \in N, a_{0n} = 1 \tag{5.8}$$

$$u_{k0} \geq c_{n0} + q_{kn} - \underbrace{(1 - x_{n0k})M}_{5.9.1}, \quad \forall n \in N, k \in K \tag{5.9}$$

$$q_{kn} \geq u_{kn} + \underbrace{sr_{kn}}_{5.10.1} - \underbrace{(1 - y_{kn})M}_{5.10.2}, \quad \forall k \in K, n \in N \tag{5.10}$$

$$q_{kn} \geq u_{kn} + w_{kn} - \underbrace{(y_{kn})M}_{5.11.1}, \quad \forall k \in K, n \in N \tag{5.11}$$

$$t_k \geq u_{k0} + \underbrace{s \sum_{n \in N} r_{kn}}_{5.12.1}, \quad \forall k \in K \tag{5.12}$$

Constraints (5.7) express the timing of the arrivals of the trains at the stations by including the travel time. This is possible only if a train travels to station n as expressed in (5.7.1). The supermarket is a specific node of the network ($n = 0$). Constraints (5.8) and (5.9) set the arrival time of the first station after the supermarket and the arrival at the supermarket after the delivery of the references. Element (5.8.1) and (5.9.1) calculate the departure and arrival time from and to the supermarket ($n = 0$) only if the vehicle travels to that node. Constraints (5.10) calculate the departure time from the stations when they are *served* by a train. The delivery time is the same for all the containers that are used for the delivery, e.g. line stocking, kitting, and stationary kitting. In (5.10.1), we consider the time to unload the containers. This occurs only if station n is *served* in (5.10.2). While constraints (5.10) consider the delivery time at the stations when they are *served*, i.e. y_{kn} is equal to 1, constraints (5.11) consider the waiting time when they are *visited*, i.e. y_{kn} is equal to 0 in (5.11.1). Constraints (5.11) are not part of the formulation of a regular IRP model since waiting times are neglected. In constraints (5.12), the roundtrip time of a tow train t_k is determined by the arrival time at the supermarket u_{k0} plus the loading time at the supermarket in (5.12.1). We assume that the replenishment activities that occur in the supermarket are done after the delivery to the stations.

Loading Constraints These constraints refer to the load and the delivery of components on the tow trains.

$$r_{kn} \geq f_n t_h - \underbrace{(1 - y_{kn})M}_{5.13.1}, \quad \forall n \in N, h, k \in K \tag{5.13}$$

$$\sum_{n \in N} (d_n r_{kn}) \leq M_k, \quad \forall k \in K \tag{5.14}$$

$$\sum_{k \in K} (d_n r_{kn}) \leq L_n, \quad \forall n \in N \tag{5.15}$$

$$v_{km} \geq v_{kn} - \underbrace{d_m r_{km}}_{5.16.1} - \underbrace{(1 - x_{nmk})M}_{5.16.2}, \quad \forall k \in K, n \in N, m \in N \tag{5.16}$$

$$\underbrace{\frac{(q_{kn} - u_{hn} - \alpha)}{M}}_{5.17.1} - \underbrace{(1 - y_{kn})}_{5.17.2} \leq e_{hn}, \quad \forall n \in N, k, h \in K \tag{5.17}$$

$$\underbrace{\frac{(\alpha + q_{hn} - u_{kn})}{M}}_{5.18.1} - \underbrace{(1 - y_{kn})}_{5.18.2} \leq b_{hn}, \quad \forall n \in N, h, k \in K \tag{5.18}$$

$$\begin{aligned} w_{hn} \leq \underbrace{(q_{kn} - u_{hn})}_{5.19.1} - \underbrace{(y_{hn})M}_{5.19.2} - \underbrace{(1 - y_{kn})M}_{5.19.3} \\ - \underbrace{(1 - e_{hn})M}_{5.19.4} - \underbrace{(1 - b_{hn})M}_{5.19.5}, \quad \forall k, h \in K, n \in N \end{aligned} \tag{5.19}$$

$$v_{k0} = \sum_{n \in N} d_n r_{kn}, \quad \forall k \in K \tag{5.20}$$

Constraints (5.13) determine the number of boxes r_{km} for each tow train (integer) by considering the longest of all roundtrip times. We use the maximum roundtrip time of the vehicles t_h so that the routes are periodic (see Figure 5.1). The big M in (5.13.1) ensures that containers are delivered only to a station that is *served* (y_{kn}). The total load of a train ($\sum_{n=1}^{N} d_n r_{kn}$) is limited to the maximum capacity of a train M_k in constraints (5.14), and the total quantity delivered to a station ($\sum_{k=1}^{K} d_n r_{kn}$) cannot exceed its capacity L_n in constraints (5.15). Constraints (5.16) determine the load v_{kn} of a vehicle k along the route. The quantity delivered at each station is subtracted from the current load of the vehicle in (5.16.1). This occurs only if the train travels to the station m in (5.16.2). This set of constraints avoids also short cycles. Constraints (5.17) and (5.18) calculate the possibility that a train must queue during *visiting* a station, encoded in decision variables e_{hm} and b_{hm}. These variables are necessary in order to understand if other trains that *visit* a station arrive earlier or later than the train that *serves* it. We need these variables to correctly calculate the waiting time when a tow train meets another one that delivers references to a station. In constraints (5.17), if a train k that *serves* a station n departs after a *visiting* train h, e_{hn} becomes equal to 1. This is done in (5.17.1) by considering the departure time of the train k and the arrival time of the train h. This occurs if the train k *serves* a station n in (5.17.2). Constraints (5.18) ensure that b_{hn} is equal to 1 if a *visiting* train h departs from the station n after the arrival of the *serving* train k. The elements (5.18.1) and (5.18.2) are similar to what is done in (5.17.1) and in (5.17.2). We assume that if the *visiting* and the *serving* train arrive at the station at the same time, the *visiting* train must wait. We use α as a lower order term that ensures that e_{kn} and b_{kn} are equal to 1 for such situations. Constraints (5.19) compute the waiting time w_{hn} for a station n for all the

trains h that do not *serve* it. (5.19.1) consists of the time between the departure of the *serving* train k and the arrival of the *visiting* train h. (5.19.2) and (5.19.3) are big M terms that ensure that the train h and k are the *visiting* and the *serving* vehicles. (5.19.4) and (5.19.5) ensure that e_{hn} and b_{hn} are equal to 1 if a vehicle must wait in a queue. Constraints (5.20) ensure that the total load v_{kn} of a vehicle k is assigned to it after it departs from the supermarket ($n = 0$).

Valid Inequalities

$$\sum_{k \in K} r_{kn} \geq 1, \quad \forall n \in N \tag{5.21}$$

Constraints (5.21) force at least one container r_{kn} to be delivered to the stations that are *served* by a train.

Variable Definition

$$\begin{aligned} x_{nmk}, e_{kn}, b_{kn} \in \{0,1\}, \\ r_{kn}, \in \mathbb{N}^+, v_{kn}, u_{kn}, q_{kn}, w_{kn}, t_k \in \mathbb{R}^+, \\ \forall k \in K, n \in N, m \in N \end{aligned} \tag{5.22}$$

Constraints (5.22) define the decision variables.

5.3.2 Heuristic Algorithm

Since the optimisation model yields excessive computation times to solve large-scale instances, we introduce a heuristic to solve larger instances. This algorithm relies on an LNS that was first introduced by Shaw (1998). This algorithm improves an initial solution by using iteratively removal and insertion operators. This algorithm is divided into three elements: the *initialisation phase*, the *improvement phase* that contains the removal and insertion operators, and the *acceptance method*. The heuristic can get stuck in a local minimum solution. To avoid this problem, the LNS framework is combined with an Iterated Local Search (ILS) that was described in Lourenço, Martin, and Stützle (2019). The ILS framework suggests that the algorithm LNS should be iterated to ensure to achieve the global optimum and avoid accepting a local minimum. The framework implemented in this algorithm is similar to the one of the ILS. In the heuristic, we define two additional parameters that are used in the notation:

- λ_k traveling time of vehicle k for the delivery of the components ($[minute]$). This time is the sum of all the time spent in traveling activities from the supermarket to the assembly stations and back.
- g_k consumption rate of the stations *served* by train k ($[\frac{box}{minute}]$). This is the sum of the consumption rates of all the stations *served* by vehicle k.

Initialisation Phase

The *initialisation phase* generates a first feasible solution for the delivery of the components to the stations. In the *improvement phase*, this solution will be improved. Therefore, this phase only aims at developing a solution that is feasible even if the objective value of this solution is far from the optimum. We define as *free stations* those stations that are not already assigned to a vehicle.

$$t_k = \lambda_k + 2sg_kt_k \tag{5.23}$$

For each vehicle k, it is possible to calculate the roundtrip time which depends on the traveling time λ_k and the demands of containers of all the stations *served* by a train k g_k. This roundtrip time can be obtained from equation (5.23). It does not consider any delay time due to congestion problems.

$$t_k = \frac{\lambda_k}{(1 - 2sg_k)} \tag{5.24}$$

Equation (5.23) can be elaborated and thus equation (5.24) is obtained. Through this equation, it is possible to identify infeasible delivery routes when the term $2sg_k$ assumes a value higher than 1. Thus, if a *free station*, increases $2sg_k$ more than the value of 1 it is not added to the route of the vehicle k.

$$r_{kn} \geq \lceil f_n t_h \rceil \tag{5.25}$$

Although equation (5.23) describes the roundtrip of each vehicle k, this time does not include the time the tow trains spend waiting in queues. Thus, it is necessary to estimate the total roundtrip time to understand the quantity that must be delivered to the stations. The quantity that must be delivered at each station n by each vehicle k can be estimated from the maximum roundtrip time of the K vehicles and is obtained from equation (5.25). We iteratively develop a delivery schedule until we can estimate

the number of containers r_{kn} that can be delivered and that can satisfy equation (5.25). At each iteration, we consider if equation (5.25) is valid. If this does not occur, we must increase the number of containers r_{kn} delivered to station n by train k. This schedule consists of a departure time from the supermarket, a traveling time to the stations, and a delivery time or a queue time.

Improvement Phase

The *improvement phase* enhances the initial solution by finding one that has a lower objective value. There are multiple removal and insertion operators. A removal operator removes an element from the route of a vehicle:

- Random Removal (RR): a station is randomly removed from the solution. This operator makes the search general so that it can select also stations that are not selected by other operators.
- Worst-distance Removal (WDR): the station that is farthest away from the previous one *served* by a vehicle is removed from the solution. This helps identify one of the stations that most increases the value of the objective function.
- Worst-station Removal (WSR): the station that requires the most time to be *served* is removed from the solution. Differently from the WDR operator that considers the travel time, the *serving* time is considered in this operator. This helps identify one of the stations that most increases the value of the objective function.
- Best-distance Removal (BDR): the station that is closest to the previous one *served* by a vehicle is removed from the solution. Congestion problems might also occur if two close stations are *served* by the same vehicle. This operator identifies such situations.
- Best-station Removal (BSR): the station that requires the least time to be *served* is removed from the solution.
- Multiple Random Removal (MRR): multiple random stations are removed from the solution.
- Zone Random Removal (ZRR): multiple random stations that are *served* by the same vehicle are removed from the solution.

An insertion operator inserts the element back into the solution in a different route:

- Closest Insertion (CI): the element is inserted in the vehicle that *serves* the closest station to the one removed. This operator reduces the objective value by reducing the traveling time of the operators.
- Random Insertion (RI): the element removed is inserted in a random vehicle. This operator is intended to be used to avoid waiting times.
- Workload Insertion (WI): the element is inserted in the vehicle that *serves* the lowest number of stations. In the IRP, a vehicle might have a high roundtrip time because it *serves* a lot of stations. This leads to a large amount of references to be delivered, which in turn increases also the roundtrip time. This operator addresses this issue.
- Exchange Insertion (EI): the element is exchanged with another element in a random vehicle. In other words, two elements from two vehicles are exchanged.

The heuristic selects one removal and one insertion operator in order to decrease the objective value. Any removal operator, insertion operator, and element are selected to ensure that the new solution is feasible. For instance, we cannot remove a single station s from a vehicle k if that vehicle *serves* only that station s so that constraints are not violated. To evaluate a solution after each removal and repair operation, we develop a delivery schedule as it was done for the *initialisation phase*. In this heuristic, we consider the minimum solution and the local solution. The former is the solution with the minimum objective value that is obtained so far, the latter is a solution that is momentarily accepted with the aim to move away from the local minimum. In order to avoid the heuristic getting stuck in a local minimum, the algorithm can perform up to θ iterations where the local solution, i.e. a less than optimum solution, can be accepted. During these θ iterations, the solutions are stored as a local solution. If after θ iterations a new minimum solution is found, then this solution is stored as the new minimum solution. Otherwise, the algorithm restarts from the last minimum solution that is found. The *stopping criteria* are to perform at least ϕ iterations which do not include the θ iterations performed to identify a local solution. Figure 5.3 provides the heuristic.

Acceptance Method

The acceptance method is the criterion used to accept a solution. In the *improvement phase*, we differentiate between a local solution and the minimum solution. A local solution is accepted if it is a feasible one. On the other hand, a minimum solution requires the feasibility of the solution and an objective value that is lower than the previous one.

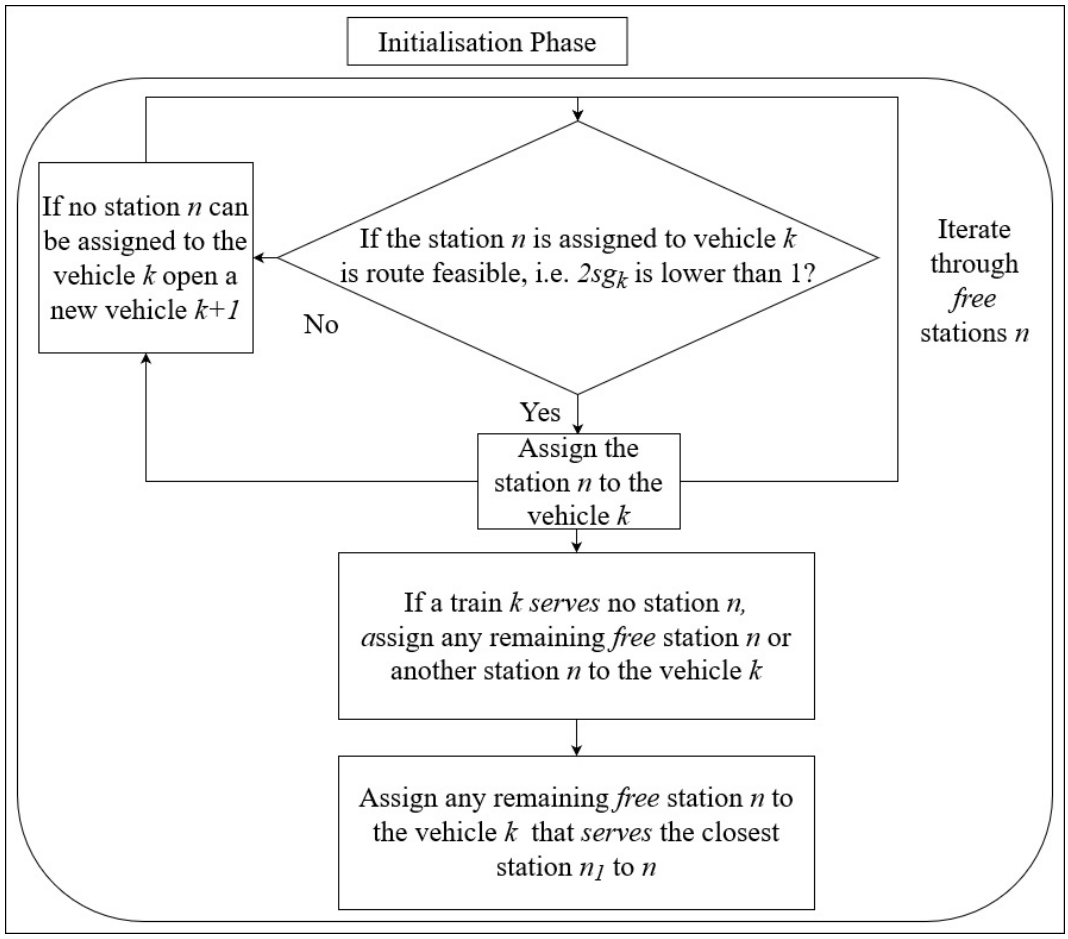
5.4 Numerical Study

In the numerical study, we first provide some Key Performance Indicators (KPIs) to quantify the quality of the results. Then, we describe the data generation process. Eventually, we validate the heuristic and we present the results that are obtained.

5.4.1 Key Performance Indicators

In order to evaluate and compare the solutions obtained by the optimisation model and the heuristic, we define five KPIs:

- The *computation time* is the time needed to solve the instances. For the optimisation model, this is the time to obtain the optimal solution. For the heuristic, this is the time after which the algorithm stops, i.e. the time to perform α iterations.
- The *total roundtrip time* is the sum of the roundtrip times of all the trains K. This is the value of the objective function of the optimisation model and the heuristic (see Section 5.3). In this case, this is the deviation between the heuristic and the optimisation model. It is also possible to calculate the *total roundtrip time* obtained with the IRP compared to the IRSP. The time deviation is the increase in the objective value due to the implementation of the IRSP.
- The *queue time* is the time spent by the tow trains while in a queue. The lower the *queue time* the more amount of time can be spent on other productive activities. The difference between the *queue time* and the *total roundtrip time* is that the first one is the time spent specifically by drivers on waiting activities. The *total roundtrip time* consists of the time of traveling time, delivery time, replenishment



initialisation phase;
store the current solution as the minimum solution;
store the current solution as the local solution;
while *stopping criteria is TRUE* **do**
    retrieve the minimum solution;
    for $i < \theta$ **do**
        use a removal operator;
        use a repair operator;
        store the current solution as the local solution;
        calculate T'_{tot} of the current solution;
        if $T'_{tot} < T_{tot}$ **then**
            store the current solution as the minimum solution;

Figure 5.3: Flow chart of the *initialisation phase* and algorithm of the heuristic.

time, and the queue time. A higher *total roundtrip time* might also be caused by a larger amount of references delivered to the stations.

- The *volume stored at the stations* is the quantity of reference kept at the stations. Usually, the higher the consumption rate and the farther a station from the supermarket is, the larger amount of references must be stored there. A lower amount of references is more desirable since it means more space available at the station for assembly or production activities.

- The *number of containers stored at the stations* is the quantity of containers that are delivered by the tow trains at the stations. The times spent on unloading, loading, and replenishing operations are dependent on the number of containers. It is also possible that there is a station where a small number of large containers are delivered.

5.4.2 Data Generation

There is no data available in the literature on the distances between the stations (a_{nn} and c_{nn}) of a manufacturing area. Thus, we collected this information from one company that produces components for the automotive industry. This company employed the process layout in a production plant. We use the data of this network to generate multiple instances for the experiments described in this section. All the instances solved in this section are related to at least one of three layouts that are common in a production environment: the process layout, fixed product layout, and product line layout (Tompkins et al. 2010). Although there might be other kinds of layouts that are less frequently employed, these three layouts represent the networks of roads that can be often found in a productive environment. The distances between the stations (c_{nm}) of the three layouts are synthetically generated based on the data collected from the company. In order to generate these distances, we employ the Descriptive Sampling technique. This approach was first described in Saliby (1980), Saliby (1990), and Saliby and Ray (1993). For these values, we sample the values based on the probability of occurrence of the distances in the range (see Saliby (1990)). As explained in Saliby (1990), we also perform a random permutation to assign the distances to the stations. The distances of the roads that connect the supermarket to other stations, c_{0m} and c_{m0}, have a different range compared to the ones that connect two stations. For these distances, we sample these values using Monte-Carlo sampling. In the model and heuristic provided in this chapter, we consider direct roads, i.e. roads that connect

two neighbouring stations. Further, we must make sense of the network of roads. When considering the roads that connect the supermarket to station n, c_{0n}, and the station n to station m, c_{nm}, the distance from the supermarket to station m, c_{0m} must be equal to the sum of the other two. This is because in many manufacturing plants the supermarket cannot be located right next to the assembly stations. In the optimisation model, we consider the arrival and departure times (u_{kn} and q_{kn}) for all the stations and the supermarket ($n = 0$). If a vehicle can travel multiple times through some stations, the index n in the set N must represent these stations, and this increases the complexity of the problem. To maintain reasonable computation times, we consider only layouts that allow the vehicles to travel only once through the stations. The company also provided the remaining parameters of the models. The consumption rates (d_n and f_n) are randomly generated using Monte-Carlo sampling. We draw a value uniformly distributed between ranges obtained from the company and based also on the author's experience. The ranges, values, and sets of the parameters of the model can be seen in Table 5.2. For the three layouts, we vary the number of stations that are *served* by the tow trains N', the number of tow trains used K', the delivery time per box s, and the demand of the stations (d_n and f_n). The maximum stock L_n at the station n and the maximum loading volume M_k are set following the author's experience and are assumed to be equal for all the stations n and vehicles k.

Table 5.2: Ranges of the parameters of the model.

Parameters of the model		
Not.	Unit	Range
c_{nm}	minute	$[0.15, 0.45]$
$c_{0m} \wedge c_{n0}$	minute	$[3.8, 4.2]$
d_n	$\frac{\text{m}^3}{\text{minute}}$	$[0.02, 0.04]$
f_n	$\frac{\text{box}}{\text{minute}}$	$[0.03, 0.22]$
L_n	m^3	0.5
M_k	m^3	21
s	$\frac{\text{minute}}{\text{box}}$	$[0.2, 0.5]$

5.4.3 Results

In this section, we first validate the heuristic by comparing its performance with the optimisation model. Then, we apply the heuristic to three large-scale instances. This

shows the magnitude of congestion problems in a manufacturing environment. The computations were performed on a personal computer with an Intel (R) Core(TM) i7-10750H CPU @2.60GHz with 32 GB of RAM (31.8 usable) and a 64-bit Windows 10. The optimisation model was realised with Xpress-IVE version 8.5, and the heuristic was realised with Python 3.8.5 64-bit coded with Spyder 4.1.5 part of the Anaconda distribution.

Validation of the Heuristic Algorithm

We validate the heuristic explained in Section 5.3.2 by comparing it to the optimisation model detailed in Section 5.3.1 for the IRSP and the IRP. For the IRSP, the model and the algorithm are compared as they are presented. For the IRP, the models do not consider any congestion problem that might occur as it is usually done for regular IRPs. In order to do this, constraints (5.17 — 5.19) are omitted from the formulation. In constraints (5.11), the waiting time for all vehicles and all stations w_{kn} is equal to 0. We solve 144 instances with a variable number of stations N' between 12 and 21. We vary the consumption rate of the stations d_n and f_n and the delivery time per container s. We solve 144 instances to validate the heuristic. In Table 5.3, we provide the computation times and the deviation between the optimisation model and the heuristic. These two KPIs can better describe the comparison between the optimisation model and the heuristic.

Table 5.3: *Computation times* of the optimisation model and the heuristic and deviation between them for 144 instances.

	Computation Times [s]				Deviation [%]	
	IRSP		IRP		IRSP	IRP
	Optimisation Model	Heuristic Algorithm	Optimisation Model	Heuristic Algorithm		
Maximum	1800.06	14.58	139.7	14.53	9.82	4.76
3rd quartile	191.07	12.98	7.59	12.95	2.54	1.07
Median	5.1	8.62	1.24	8.81	1.05	0.0
Average	274.53	9.7	6.72	9.76	1.8	0.75
1st quartile	3.09	7.13	0.8	7.33	0.0	0.0
Minimum	2.06	5.84	0.43	6.03	0.0	0.0

The results show that the heuristic can find a near-optimal solution in a shorter amount

of time than the optimisation model. On the other hand, the optimisation model is able to solve the IRSP and the IRP in a shorter period of time in the minimum case. This is because the heuristic requires at least some time to improve the initial solution. For some instances of the IRSP, the computation times might become prohibitive. The computation times are higher for the IRSP than for the IRP for the optimisation model. The reason for this is that to solve the IRSP, it is necessary to have a higher number of decision variables (e_{kn}, b_{kn}, and w_{kn}) to account for the congestion problems. On the other hand, the heuristic requires the same amount of time for both the IRSP and the IRP. The deviation between the optimisation model and the heuristic is quite low for the IRP since the maximum is lower than 5%. On the other hand, the deviation for IRSP is higher but still lower than 10%.

Comparison between the IRSP and the IRP

We solve 54 large-scale instances with a variable number of vehicles used for the delivery of the components and a number of stations N' that lies between 45 and 60. We use the heuristic to solve such large-scale instances. For the IRSP, we implement the heuristic as it is described in Section 5.3.2. For the IRP, we avoid any consideration about the congestion problems that might occur. Therefore, for the IRP, when we run the heuristic in Section 5.3.2, we do not consider if a station is being *served* when another train *visits* it. Then, a new schedule is developed for the IRSP (i.e. by considering congestion problems) with the solution obtained from the heuristic for the IRP. This shows the effects of congestion problems on the delivery route of the vehicles. Table 5.4 provides all the KPIs described in Section 5.4.1 in order to have a fair comparison between the IRP and the IRSP. The results show that there is a lower number of *containers stored at the station*, *volume stored at the stations*, and a lower *queue time* for the IRSP compared to the IRP. This is because the IRSP model can forecast the *waiting time* of the vehicles, and can reduce the *total roundtrip time*. Since the maximum *roundtrip time* decreases, so does the amount of components stored at the stations. The maximum *queue time* decreases as well although in average the decrease is only marginal. One additional benefit that is obtained from the implementation of the IRSP is that although the *queue time* is not equal to zero, the model can predict that a delay will occur. Thus, the safety stock and the quantity of components stored at the stations can be adapted based also on the *queue time*. The decrease in the *roundtrip time* and the *queue time* come also with the benefit that the drivers of the tow trains can support the picking activities at the supermarket between one roundtrip and the other. In the

objective function (5.1), we minimise the traveling time after the departure from the supermarket. With the IRSP, the *queue time* decreases and the drivers can spend more time at the supermarket before the departure.

Table 5.4: *Containers stored at the stations*, *volume stored at the stations*, *queue time*, *computation times*, and *time deviation* of the optimisation model and the heuristic for 54 large-scale instances. The time deviation refers to the increase of the objective value due to the implementation of the IRP rather than the IRSP.

	Containers stored [#]		Volume stored [m^3]		Queue time [%]	
	IRSP	IRP	IRSP	IRP	IRSP	IRP
Maximum	810	810	27.62	27.62	47.72	55.7
3rd quartile	348.5	513.0	12.34	17.85	28.93	34.57
Median	172.5	241.5	6.12	8.55	10.55	10.5
Average	271.76	328.04	9.47	11.42	15.21	16.8
1st quartile	106.0	116.75	3.76	4.11	0.34	0.36
Minimum	49	49	1.75	1.75	0.0	0.0

	Computation Times [s]		Time Deviation [%]
	IRSP	IRP	
Maximum	108.86	110.92	78.02
3rd quartile	95.64	96.03	24.1
Median	70.31	69.17	8.78
Average	72.45	71.22	16.5
1st quartile	52.14	51.37	0.08
Minimum	45.31	44.81	0.0

This time can be allocated to support the picking activities. When scheduling the delivery with a regular IRP, the amount of time spent in queues is unknown and the managers must rely on rules of thumb to predict how much time will be wasted on queues. The *computation times* are equal between the optimisation model and the heuristic. This is because the heuristic must develop a schedule for both the IRP and the IRSP. The *time deviation* shows that the *total roundtrip time* can be decreased, on average, by 16.5% with the implementation of the IRSP rather than the IRP. This is because the IRP develops only basic schedules that avoid any considerations on congestion problems and time spent by vehicles queuing. In other words, the IRP develops schedules that include congestion problems and increases the roundtrip time. For this reason, the IRP can only decrease the traveling, loading, and unloading time and cannot identify the queuing time of the vehicles. Also here, we see the benefit that the 16.5% of the roundtrip time can instead be spent on picking activities by the operators. This means not only reducing the queue time of the drivers but also as a

benefit for the logistic operations.

Sensitivity Analysis of the Layout Types

Three major layouts are common in a production environment: the process layout, fixed product layout, and product line layout (Tompkins et al. 2010). The fixed product layout is usually implemented for the assembly of products that cannot be moved, e.g. aeroplanes or sophisticated large machinery. Due to the size or the precision required for the assembly, the product must remain still and all assembly operations are carried out around it. The product line layout is usually implemented for the assembly of cars or products in high volumes, e.g. laptops and bikes. In this case, the assembly activities are carried out on the side of the product. The process layout consists of grouping equipment and machinery that perform similar operations. This is because these tools might require similar spare parts and lubricants. The product family layout which is also listed in Tompkins et al. (2010) employs a network similar to the one found on the process layout. One could argue that only the product line layout is found in the automotive sector and that this comparison is not beneficial. Indeed, the product line layout is commonly implemented in the automotive sector, but it is also true that the other two layouts are implemented in its satellite industry. For instance, the fixed product layout is implemented for the assembly of engines of vehicles since most of the components are small and the engine must be still to ensure high efficiency. Also, companies that produce car components employ the process layout to improve efficiency. For instance, some OEMs employ the process layout for the production of car seats, exhaustion systems, and lighting system. This is because each process requires similar equipment and operations.
We solve 18 large-scale instances for each of these three layouts in order to understand which of them leads to higher congestion problems. For each layout, we solve 3 instances with 45 assembly stations N' and 3 instances with 60 assembly stations N'. Figure 5.4 depicts the networks of the roads that are implemented in the three layouts. For the IRSP, we implement the heuristic as it is described in Section 5.3.2. For the IRP, we avoid any consideration about the congestion problems that might occur during the delivery and solve the instances with the heuristic in Section 5.3.2.

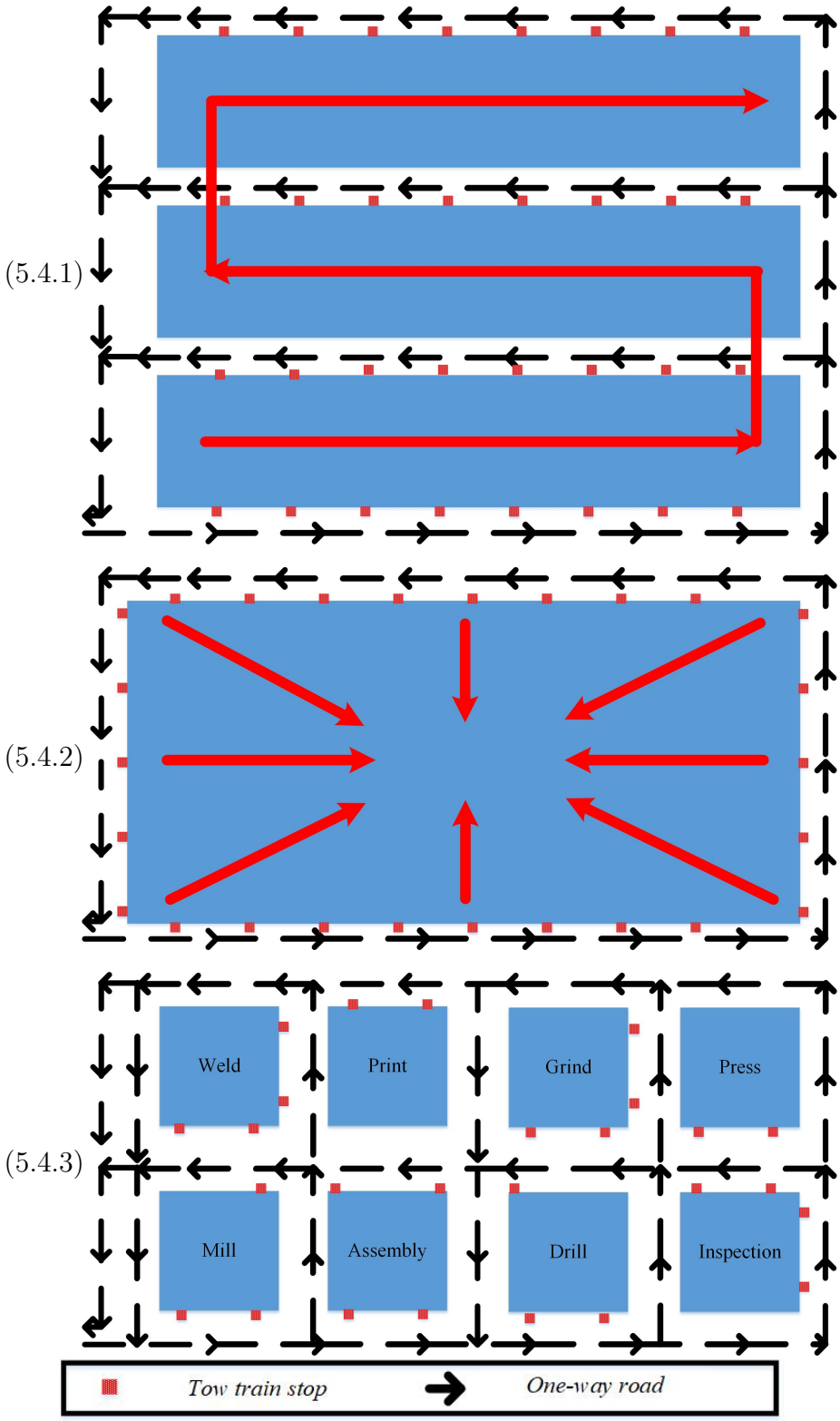


Figure 5.4: Networks of the roads that are employed in a product line layout (1), fixed product layout (2), and process layout (3).

Then, a new schedule is developed for the IRSP (i.e. by considering congestion problems) with the solution obtained from the heuristic for the IRP. Table 5.5 provides the *number of components stored at the stations*, the *volume of components stored at the stations*, and the *cost deviation*.

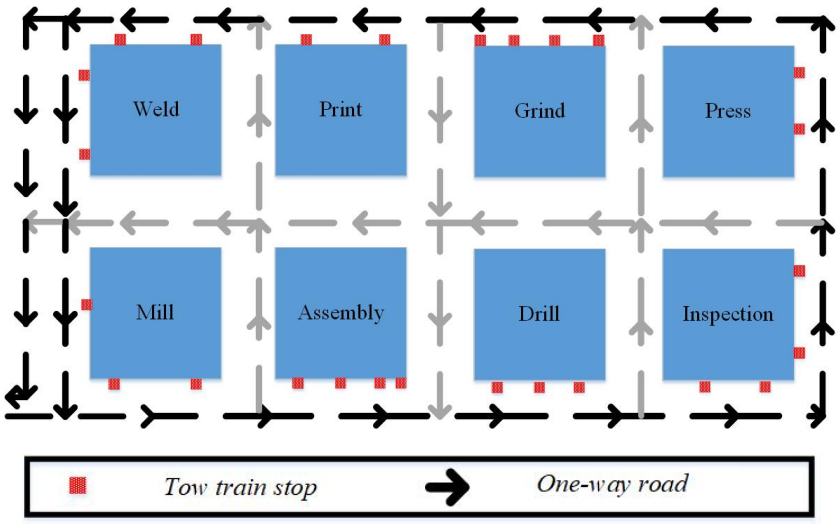


Figure 5.5: Improved process layout

The results show that the fixed product layout leads to a minimum difference between the IRSP and the IRP for the *containers stored at the stations*, the *volume stored at the stations*, and the *time deviation*. On the other hand, the process layout is the one where there is a maximum difference between the IRSP and the IRP in the *containers stored at the stations*, the *volume stored at the stations*, and the *time deviation*. Eventually, the product line layout has a difference between the IRSP and the IRP for the *containers stored at the stations*, the *volume stored at the stations*, and the *time deviation* that lies between the fixed product layout and the process layout. Thus, we can expect that those companies that employ a process layout, e.g. OEMs that produce components for the automotive sector, are more likely to suffer from congestion problems.

Table 5.5: *Container stored*, *volume stored*, and *time deviation* of the optimisation model and the heuristic for 18 large-scale instances of three main layouts.

	Containers stored [#]		Volume stored [m^3]		Time deviation [%]
	IRSP	IRP	IRSP	IRP	
Product line layout					
Maximum	810	810	27.62	27.62	69.21
3rd quartile	524.75	633.0	18.09	21.63	25.69
Median	245.5	388.5	8.69	13.74	12.85
Average	329.11	414.11	11.46	14.35	17.83
1st quartile	152.5	203.75	5.42	7.24	2.51
Minimum	98	98	3.44	3.44	0.0
Fixed product layout					
Maximum	795	556	26.99	19.59	12.1
3rd quartile	204.25	204.25	7.3	7.3	0.23
Median	94.5	98.0	3.39	3.44	0.07
Average	175.0	168.33	6.14	5.97	1.52
1st quartile	67.0	67.0	2.4	2.4	0.0
Minimum	49	49	1.75	1.75	0.0
Process layout					
Maximum	810	810	27.62	27.62	78.02
3rd quartile	498.5	631.5	17.48	21.56	46.22
Median	190.5	344.0	6.73	12.22	22.0
Average	311.17	401.67	10.82	13.93	30.17
1st quartile	120.75	183.25	4.33	6.48	9.03
Minimum	71	71	2.6	2.6	0.0

Congestion problems are less likely to occur in the environments for the assembly of engines that employs a fixed product layout. An additional insight for managers is to choose, where possible, a layout that reduces congestion problems. As well, modifying the network of roads might also be beneficial in reducing congestion problems. For instance, the process layout proposed in Figure 5.5 could be a valid substitute for the process layout in Figure 5.4.3. Notice that the internal roads are not needed anymore. Though, this new layout can be implemented only if the tow train stops and the station racks are relocated. Although reorganising the layout might not always be possible, by doing this the space available for productive operations would increase, and the layout would cause fewer congestion problems.

5.5 Conclusion

In this chapter, we discuss congestion problems that occur in a manufacturing environment during the part feeding practices. The periodic IRSP was defined. An optimisation model was developed for the IRSP. This model includes a balancing constraint and an additional objective function. Due to the long computation times of the optimisation model, a heuristic for solving problems in real-life scenarios was developed. The results of the heuristic are validated compared to the results of the optimisation model and we find the average deviation is lower than 2%. We find that the average time reduction that can be achieved if congestion problems are considered is 16.5% for large-scale instances. This time can be spent by drivers at the supermarket helping or supporting the picking activities. We perform a sensitivity analysis where we analyse the congestion problems that can occur in different manufacturing layouts. Through this analysis, we find that congestion problems are most likely to occur and affect the delivery schedule in the process layout.

A limitation of our model is that the optimisation model contains many Big-M constraints. This makes the current formulation one with a weak relaxation that is hard to solve. To limit this problem, we set the values of the Big-M large enough to ensure that the constraints are valid and no solution is mistakenly eliminated. At the same time, we set the values of Big-M small enough to ensure that the relaxation of the MILP is as small as possible. Although this helps the solver find the optimal solution, other formulations could be implemented to solve the IRSP. Further research can also provide additional formulations to solve the problem with a lower level of detail. A lower level of detail can be beneficial in reducing the number of Big-M constraints.

Although considering the IRSP from the deterministic point of view decreases the complexity of the problem, the real problem and its results are limited and subject to stochastic issues that might occur in real life. Therefore, a stochastic optimisation model can be developed for the IRSP. This would allow the development of models that adapt better to real-life situations.

Chapter 6

Conclusion

6.1 Summary of Insights

In the manufacturing sector, assembly environments are common. In many assembly environments, components are delivered to the assembly stations so that the assembly operations can be performed. These are expensive activities that are performed continuously that can negatively affect the costs of the final product. A large amount of money can be saved by companies if these activities can be improved. Through optimisation tools and machine learning techniques, we investigated these activities to improve their implementation.

In Chapter 3, we proposed an optimisation model for the ALFP. The optimisation model selects the most appropriate line feeding mode for each component to minimise the total costs for the delivery of all the components. We used a CART algorithm to develop a decision tree that allows to identify the attributes of the components that affect the selection of the line feeding mode. This decision tree can explain the decision for the line feeding mode selection and predict with a classification accuracy of 78.49% the line feeding modes of the components. We trained the decision tree with data synthetically generated from data obtained from multiple companies from different industries. Although a line feeding mode is assigned by the decision tree to each component based on its attributes, there is the possibility that the constraints of the optimisation model are violated. We developed a repair approach that ensures that the solution obtained from the implementation of the decision tree is feasible and that reduces the cost deviation from the optimal solution. We found that the average cost deviation after the repair approach is 0.38%.

In Chapter 4, we studied the JALBFP which is a combination of the ALBP and the ALFP. This problem jointly considers how the tasks are assigned to the stations and

which line feeding mode is assigned to the components. Companies usually solve the ALBP during the planning stage of an assembly line and the ALFP when the assembly line is operated and the components must be assigned to a line feeding mode. We developed an optimisation model and a heuristic algorithm that relies on an ALNS for the JALBFP. We validated the results obtained from the heuristic algorithm and the optimisation model. We found that the JALBFP leads to an average total cost reduction of 13.84% compared to a sequential approach that is usually implemented in a productive environment. We found that the reason for the total cost reduction is that the sequential approach has higher transportation and supermarket costs. In a sensitivity analysis, we found that a higher number of components, a higher total volume of the components, and a lower space at the BoL lead to a higher cost reduction.
In Chapter 5, we studied the delivery of the components to an assembly line performed with tow trains. Companies usually employ a model for the IRP that routes the vehicles to the stations without any consideration of congestion problems that might occur. We proposed an optimisation model and a heuristic algorithm for the IRSP that combines the IRP with the SSP so that congestion problems and routing decisions can be jointly considered. These approaches can be used to avoid congestion problems and queues that might occur during the delivery of the components. We proposed a numerical study based on data obtained from a company. We found that congestion problems could lead to an increase in the delivery time of 16.5% if standard routing approaches are implemented for the delivery of the components. In the numerical study, we proposed a sensitivity analysis to identify the layout that is most likely to lead to congestion problems and find that the process layout is the one that is most likely to lead to congestion problems.

6.2 Agenda for Future Research

For the ALFP in Chapter 3, we developed an optimisation model that considers the *stationary kit* as a line feeding mode rather than the *traveling kit*. This decision was made because we wanted to consider realistic cases for the *training*, *validation*, and *testing* samples of the decision tree and therefore we could only consider the line feeding modes that were implemented in those industrial environments. A limitation of this work is that we considered the *stationary kit* as a line feeding mode, but other industrial environments could implement the *traveling kit*. A decision tree could be implemented to provide some guidelines on which line feeding mode between the two

is more suitable for each industry or plant based on their attributes. This could provide a clear answer on the conditions that make one of the two line feeding modes most appropriate and economically convenient based on the space and environment. Alternatively, future research should consider the *traveling kit* rather than the *stationary* one in the development of a decision tree. This can provide valuable insights into the implementation of *traveling kits*.

For the JALBFP, the model explained in Chapter 4 combines the ALBP and the ALFP. In order to be able to solve large-scale instances, a heuristic algorithm was presented. Our formulation considered already a high level of detail for a multi-manned assembly line and it is not possible to consider also the ergonomic workload of the assembly operations. Future research should investigate the ergonomic implications of this problem by considering the energy expenditure of the assembly and line feeding activities or the fatigue level. The ergonomic limits to this problem should be considered to avoid uncomfortable situations for the workers through the use of constraints or through the use of an objective function that minimises the workload. In other words, ergonomic consideration can help assign the tasks to the stations so that the ergonomic workload of the operations in the assembly line and the line feeding mode selection are feasible for the operator. Considering the ergonomic workload of the operations could help to more realistically plan a multi-manned assembly line and the operations that are performed there.

In Chapter 5, we developed an optimisation model and a heuristic algorithm to solve the IRSP. One of the limitations of this work is that the model and the algorithm analyse the problem from a deterministic point of view. Future research should investigate the delivery of the components from a stochastic point of view. Specifically, this study should consider stochastic traveling times for the tow trains, stochastic consumption rates for the station, and stochastic loading and unloading times. With this technique, a more realistic schedule for the delivery of the components can be developed. On the other hand, the heuristic algorithm that was developed allows already to recreate the delivery of the references. The new heuristic algorithm should consider stochastic times in the delivery but can preserve the scheduling technique described in Section 5.3.2.

Bibliography

Abdul, R., M. Kamarul Irwan, Y. Zhong, E. Aghezzaf, and T. Aouam. 2014. "Modelling and Solving the Multiperiod Inventory-Routing Problem with Stochastic Stationary Demand Rates." *International Journal of Production Research* 52 (14): 4351–4363.

Akpinar, S., A. Elmi, and T. Bektaş, T. 2017. "Combinatorial Benders cuts for assembly line balancing problems with setups." *European Journal of Operational Research* 259 (2): 527-537.

Aksen, D., O. Kaya, F. S. Salman, and Y. Akça. 2012. "Selective and Periodic Inventory Routing Problem for Waste Vegetable Oil Collection." *Optimization Letters* 6 (6): 1063–1080.

Aksen, D., O. Kaya, F. S. Salman, and Ö. Tüncel. 2014. "An Adaptive Large Neighborhood Search Algorithm for a Selective and Periodic Inventory Routing Problem." *European Journal of Operational Research* 239 (2): 413–426. .

Altman, N. S. 1992. "An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression." *The American Statistician* 46 (3): 175.

Álvarez-Miranda, E., S. Chace, and J. Pereira. 2021. "Assembly line balancing with parallel workstations." *International Journal of Production Research* 59 (21): 1-21.

Anderson, J. 2020. "Charging Up the Automotive Industry." *Porsche Consulting The Magazine*, September 7 2020. `https://magazine2020.porsche-consulting.com/en/car-industry/`

Andersson, H., M. Christiansen, and G. Desaulniers. 2015. "A New Decomposition Algorithm for a Liquefied Natural Gas Inventory Routing Problem." *International Journal of Production Research* 54 (2): 564–578.

"Annual Report 2020". 2021. *Stellantis N.V.* `https://www.stellantis.com/content/dam/stellantis-corporate/investors/financial-reports/Stellantis_2020_12_31_Annual_Report.pdf`

"Annual Report 2020." 2021. *Volkswagen AG* `https://annualreport2020.volkswagenag.com/servicepages/createpdf_action.php?ranges%5B%5D=1-371`

Baller, R., S. Hage, P. Fontaine, and S. Spinler. 2020. "The Assembly Line Feeding Problem: An Extended Formulation with Multiple Line Feeding Policies and a Case Study." *International Journal of Production Economics* 222: 107489.

Bard, J. F., and N. Nananukul. 2009. "Heuristics for a Multiperiod Inventory Routing Problem with Production Decisions." *Computers & Industrial Engineering* 57 (3): 713–723.

Bartlett, K., J. Lee, S. Ahmed, G. Nemhauser, J. Sokol, and B. Na. 2014. "Congestion-aware Dynamic Routing in Automated Material Handling Systems." *Computers & Industrial Engineering* 70: 176-182.

Battini, D., N. Boysen, and S. Emde. 2013. "Just-in-Time Supermarkets for Part Supply in the Automobile Industry." *Journal of Management Control* 24 (2): 209-217.

Battini, D., M. Calzavara, S. Mazzon, and F. Sgarbossa. 2016. "Balancing Assembly Line and Part Picking at Once to Achieve Economic Gains in Assembly System Management." *Proceedings of the 19th International Working Seminar on Production Economics*, 22-26 February 2016, Innsbruck, Austria.

Battini, D., M. Calzavara, A. Otto, and F. Sgarbossa. 2017. "Preventing Ergonomic Risks with Integrated Planning on Assembly Line Balancing and Parts Feeding." *International Journal of Production Research* 55 (24): 7452–7472.

Battini, D., M. Faccio, A. Persona, and F. Sgarbossa. 2009. "Design of the Optimal Feeding Policy in an Assembly System." *International Journal of Production Economics* 121 (1): 233–254.

Battini, D., M. Gamberi, A. Persona, and F. Sgarbossa. 2015. "Part-Feeding with Supermarket in Assembly Systems: Transportation Mode Selection Model and Multi-Scenario Analysis." *Assembly Automation* 35 (1): 149–159.

Bell, W. J., L. M. Dalberto, M. L. Fisher, A. J. Greenfield, R. Jaikumar, P. Kedia, R. G. Mack, and P. J. Prutzman. 1983. "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer." *Interfaces* 13 (6): 4–23.

Bertsimas, D., and J. Dunn. 2017. "Optimal Classification Trees." *Machine Learning* 106 (7): 1039–1082.

Bischl, B., M. Lang, L. Kotthoff, J. Schiffner, J. Richter, E. Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. "mlr: Machine Learning in R." *The Journal of Machine Learning Research* 17 (1): 5938-5942.

Boudella, M. E. A., E. Sahin, and Y. Dallery. 2018. "Kitting optimisation in Just-in-Time mixed-model assembly lines: assigning parts to pickers in a hybrid robot–operator kitting system." *International Journal of Production Research* 56 (16): 5475-5494.

Bowman, E. H. 1960. "Assembly-Line Balancing by Linear Programming." *Operations Research* 8 (3): 385–389.

Boysen, N., M. Fliedner, and A. Scholl. 2008. "Assembly Line Balancing: Which Model to Use When?" *International Journal of Production Economics* 111 (2): 509–528.

Boysen, N., M. Fliedner, and A. Scholl. 2009. "Assembly Line Balancing: Joint Precedence Graphs under High Product Variety." *IIE Transactions* 41 (3): 183–193.

Bozer, Y. A., and L. F. McGinnis. 1992. "Kitting versus Line Stocking: A Conceptual Framework and a Descriptive Model." *International Journal of Production Economics* 28 (1): 1–19.

Breiman, L. 1984. *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

Breiman, L. 2001. "Random forests." *Machine Learning* 45 (1): 5-32.

Burkacky, O., S. Lingemann, and K. Pototzky. 2021. "Coping With the Auto-Semiconductor Shortage: Strategies for Success." *McKinsey & Company* `https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/coping-with-the-auto-semiconductor-shortage-strategies-for-success`

Calzavara, M., S. Finco, D. Battini, F. Sgarbossa, and A. Persona. 2021. "A Joint Assembly Line Balancing and Feeding Problem (JALBFP) Considering Direct and Indirect Supply Strategies." *International Journal of Production Research.* Advance online publication. doi: 10.1080/00207543.2021.1968527.

Camci, F., R. B. Chinnam, and R. D. Ellis. 2008. "Robust Kernel Distance Multivariate Control Chart Using Support Vector Principles." *International Journal of Production Research* 46 (18): 5075–5095.

Campbell, A. M., and M. W. Savelsbergh. 2004. "A Decomposition Approach for the Inventory-Routing Problem." *Transportation Science* 38 (4): 488–502.

Caputo, A. C., and P. M. Pelagagge. 2011. "A Methodology for Selecting Assembly Systems Feeding Policy." *Industrial Management & Data Systems* 111 (1): 84–112.

Caputo, A. C., P. M. Pelagagge, and P. Salini. 2015. "A Decision Model for Selecting Parts Feeding Policies in Assembly Lines." *Industrial Management & Data Systems* 115 (6): 974–1003.

Caputo, A. C., P. M. Pelagagge, and P. Salini. 2018. "Selection of Assembly Lines Feeding Policies Based on Parts Features and Scenario Conditions." *International Journal of Production Research* 56 (3): 1208–1232.

Caputo, A. C., P. M. Pelagagge, and P. Salini. 2021. "A model for planning and economic comparison of manual and automated kitting systems." *International Journal of Production Research* 59 (3): 885-908.

Celisse, A., and S. Robin. 2008. "Nonparametric Density Estimation by Exact Leave–out Cross-Validation." *Computational Statistics & Data Analysis* 52 (5): 2350–2368.

Cerqueus, A., and X. Delorme. 2019. "A Branch-and-Bound Method for the Bi-objective Simple Line Assembly Balancing Problem." *International Journal of Production Research* 57 (18): 5640-5659.

Chan, T.F., G.H. Golub, and R.J. LeVeque. 1982. "Updating Formulae and a Pairwise Algorithm for Computing Sample Variances." *COMPSTAT 1982 5th Symposium Held at Toulouse 1982*: 30–41.

Chiew, K. 2012. "Scheduling and routing of autonomous moving objects on a mesh topology." *Operational Research* 12 (3): 385-397.

Chiew, K., and S. Qin. 2009. "Scheduling and Routing of AMOs in an Intelligent Transport System." *IEEE Transactions on Intelligent Transportation Systems* 10 (3): 547-552.

Çil, Z.A., and Kizilay, D. 2020. "Constraint Programming Model for Multi-manned Assembly Line Balancing Problem." *Computers & Operations Research* 124: 105069.

Coelho, L. C., and G. Laporte. 2013. "A Branch-and-Cut Algorithm for the Multi-Product Multi-Vehicle Inventory-Routing Problem." *International Journal of Production Research* 51 (23-24): 7156–7169.

Cortes, C., and V. Vapnik. 1995. "Support-vector networks." *Machine Learning* 20 (3) : 273-297.

Dantzig, G. B., and J. H. Ramser. 1959. "The Truck Dispatching Problem." *Management Science* 6 (1): 80–91.

Dimitriadis, S. G. 2006. "Assembly Line Balancing and Group Working: A Heuristic Procedure for Workers' Groups Operating on the Same Product and Workstation." *Computers & Operations Research* 33 (9): 2757–2774.

Dixon, W. J. 1950. "Analysis of Extreme Values." *The Annals of Mathematical Statistics* 21 (4): 488-506.

Elliott, R. 2021. "Tesla Faces New Reality of Tough Competition." *Wall Street Journal*, March 11 2021. `https://www.wsj.com/articles/tesla-faces-new-reality-of-tough-competition-11615478406`

Emde, S., and N. Boysen. 2011. "Optimally Routing and Scheduling Tow Trains for JIT-Supply of Mixed-Model Assembly Lines." *European Journal of Operational Research* 217 (2): 287-299.

Efron, B., and R. J. Tibshirani. 1993. *An Introduction to the Bootstrap.* New York: Chapman and Hall.

Fanti, M. P. 2002. "Event-based controller to avoid deadlock and collisions in zone-control AGVS." *International Journal of Production Research* 40(6): 1453-1478.

Farrell, D., J. Remes, V. Agrawal, and et al. 2003. "Preface to the Auto Sector Cases." In "New Horizons: Multinational Company Investment in Developing Economies." *McKinsey & Company* `https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/new-horizons-for-multinational-company-investment`

Fletcher, R. 2008. *Practical Methods of Optimization.* Chichester: Wiley.

Fransen, K. J. C., J. A. W. M. Van Eekelen, A. Pogromsky, M. A. Boon, and I. J. Adan. 2020. "A Dynamic Path Planning Approach for Dense, Large, Grid-based Automated Guided Vehicle Systems." *Computers & Operations Research* 123: 105046.

Freund, H., L. Jánoskuti, A. Kadocsa, D. Svoboda, and A. Tschiesner. 2020. "Rethinking European Automotive Competitiveness: The R&D CEE Opportunity." *McKinsey & Company* https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/rethinking-european-automotive-competitiveness-the-r-and-d-cee-opportunity

Gaur, V., and M. L. Fisher. 2004. "A Periodic Inventory Routing Problem at a Supermarket Chain." *Operations Research* 52 (6): 813–822.

Giglio, D., M. Paolucci, A. Roshani, and F. Tonelli. 2017. "Multi-manned Assembly Line Balancing Problem with Skilled Workers: a new Mathematical Formulation." *IFAC-PapersOnLine* 50 (1): 1211-1216.

Gini, C. 1912. "Variabilità e Mutabilità." *Reprinted in Memorie di metodologica statistica.*(Ed. Pizetti E, Salvemini, T) Rome: Libreria Eredi Virgilio Veschi.

Gokgoz, E., and A. Subasi. 2015. "Comparison of Decision Tree Algorithms for EMG Signal Classification Using DWT." *Biomedical Signal Processing and Control* 18: 138–144.

Hemminki, J., T. Leipala, and O. Nevalainen. 1998. "On-Line Packing with Boxes of Different Sizes." *International Journal of Production Research* 36 (8): 2225–2245.

Hehl, B., C. Gall, J. Buss, A. Abuja, and K. Rebbereh. 2021. "How Auto Suppliers can Navigate EV Technology Disruptions in Four Steps." *EY Parthenon* https://www.ey.com/en_us/strategy/how-auto-suppliers-can-navigate-ev-technology-disruption-in-four-steps

Hofstätter, T., M. Krawina, B. Mühlreiter, S. Pöhler, and A. Tschiesner. 2020. "Reimagining the Auto Industry's Future: It's Now or Never." *McKinsey & Company* https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/reimagining-the-auto-industrys-future-its-now-or-never

International Organization for Standardization. 2003. *Ergonomics – Manual handling – Part 1: Lifting and Carrying.* ISO Standard No. 11288-1:2003. https://www.iso.org/standard/26520.html

James, G., D. W., T. Hastie, and R. Tibshirani. 2017. *An Introduction to Statistical Learning: with Applications in R.* New York: Springer.

Jarvis, D. 2020. "The AI Talent Shortage isn't Over yet." *Deloitte & Touche LLP* https://www2.deloitte.com/uk/en/insights/industry/technology/ai-talent-challenges-shortage.html

Johnson, M. E. 2001. "Modelling Empty Vehicle Traffic in AGVS Design." *International Journal of Production Research* 39 (12): 2615–2633.

Kammoun, M. A., N. Rezg, and Z. Achour. 2014. "New approach for air traffic management based on control theory." *International Journal of Production Research* 52 (6): 1711-1727.

Kim, Y. K., J. Y. Kim, and Y. Kim. 2006. "An Endosymbiotic Evolutionary Algorithm for the Integration of Balancing and Sequencing in Mixed-model U-lines." *European Journal of Operational Research* 168 (3): 838-852.

Knerr, S., L. Personnaz, and G. Dreyfus. 1990. "Single-layer Learning Revisited: a Stepwise Procedure for Building and Training a Neural Network." *Neurocomputing* 41-50. Berlin, Heidelberg: Springer.

Küpper, D., K. Kuhlmann, K. Tominaga, A. Arora, and J. Schlageter. 2020. "Shifting Gears in Auto Manufacturing." *Boston Consulting Group* `https://www.bcg.com/publications/2020/transformative-impact-of-electric-vehicles-on-auto-manufacturing`

Li, Z., Kucukkoc, I. and Tang, Q. 2020. "A Comparative Study of Exact Methods for the Simple Assembly Line Balancing Problem." *Soft Computing* 24 (15):11459-11475.

Limère, V., H. Van Landeghem, and M. Goetschalckx. 2015. "A Decision Model for Kitting and Line Stocking with Variable Operator Walking Distances." *Assembly Automation* 35 (1): 47–56.

Limère, V., H. Van Landeghem, M. Goetschalckx, E. Aghezzaf, and L. F. Mcginnis. 2012. "Optimising Part Feeding in the Automotive Assembly Industry: Deciding between Kitting and Line Stocking." *International Journal of Production Research* 50 (15): 4046–4060.

Lourenço, H. R., O. C. Martin, and T. Stützle. 2019. Iterated Local Search: Framework and Applications. *Handbook of Metaheuristics* 129-168. Cham: Springer.

Martignago, M., O. Battaïa, D. and Battini. 2017. "Workforce Management in Manual Assembly Lines of Large Products: a Case Study." *IFAC-PapersOnLine* 50 (1): 6906-6911.

McMullen, P. R., and G. V. Frazier. 1998. "Using Simulated Annealing to Solve a Multiobjective Assembly Line Balancing Problem with Parallel Workstations." *International Journal of Production Research* 36 (10): 2717–2741.

Michels, A.S., T.C. Lopes, C.G.S. Sikora, and L. Magatão. 2019. "A Benders' Decomposition Algorithm with Combinatorial Cuts for the Multi-manned Assembly Line Balancing Problem." *European Journal of Operational Research* 278 (3): 796-808

Moon, I., R. Logendran, and J. Lee. 2009. "Integrated Assembly Line Balancing with Resource Restrictions." *International Journal of Production Research* 47 (19): 5525-5541.

Müllerklein, D., P. Fontaine, and F. Ostermeier. 2022. "Integrated consideration of assembly line scheduling and feeding: A new model and case study from the automotive industry." *Computers & Industrial Engineering*: 108288.

Naderi, B., A. Azab, and K. Borooshan. 2019. "A Realistic Multi-manned Five-sided Mixed-model Assembly Line Balancing and Scheduling Problem with Moving Workers and Limited Workspace." *International Journal of Production Research* 57 (3): 643-661.

Nakasuka, S., and T. Yoshida. 1992. "Dynamic Scheduling System Utilizing Machine Learning as a Knowledge Acquisition Tool." *International Journal of Production Research* 30 (2): 411–431.

Nourmohammadi, A., and H. Eskandari. 2017. "Assembly Line Design Considering Line Balancing and Part Feeding.", *Assembly Automation* 37 (1): 135-143.

Otto, A., C. Otto, and A. Scholl. 2013. "Systematic Data Generation and Test Design for Solution Algorithms on the Example of SALBPGen for Assembly Line Balancing." *European Journal of Operational Research* 228 (1): 33–45.

Pham, D. T., and A. A. Afify. 2005. "Machine-learning Techniques and their Applications in Manufacturing." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219 (5): 395-412.

Pinto, A. M., L. F. Rocha, and A. P. Moreira. 2013. "Object Recognition Using Laser Range Finder and Machine Learning Techniques." *Robotics and Computer-Integrated Manufacturing* 29 (1): 12–22.

Qian, X., and Q. Fan. 2011. "Solving Multi-manned Assembly Line Balancing Problem by a Heuristic-mixed Genetic Algorithm." *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, 3: 320-323.

Ramachandran, K., and J. Watson. 2021. "Tech Looks to Analytics Skills to Bolster its Workforce." *Deloitte & Touche LLP* `https://www2.deloitte.com/global/en/insights/industry/technology/data-analytics-skills-shortage.html`

Richards, G. 2014. *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. 3rd ed. London: Kogan Page

Rincon-Garcia, N., B.J. Waterson, and T.J. Cherrett. 2017. "A Hybrid Metaheuristic for the Time-Dependent Vehicle Routing Problem with Hard Time Windows." *International Journal of Industrial Engineering Computations* 8 (1): 141–160.

Robinson, J. 1949. "On the Hamiltonian Game (A Traveling Salesman Problem)." *Project Rand*. Santa Monica, CA: The Rand Corporation (RM-303).

Ropke, S., and D. Pisinger. 2006. "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows." *Transportation Science* 40 (4): 455–472.

Roshani, A., and D. Giglio. 2016. "Simulated Annealing Algorithms for the Multi-Manned Assembly Line Balancing Problem: Minimising Cycle Time." *International Journal of Production Research* 55 (10): 2731–2751.

Roshani, A., and D. Giglio. 2020. "A Tabu Search Algorithm for the Cost-oriented Multi-manned Assembly Line Balancing Problem." *International Journal of Industrial Engineering & Production Research* 31 (2): 189-202.

Roshani, A., and Nezami, F.G. 2017. "Mixed-model Multi-manned Assembly Line Balancing Problem: a Mathematical Model and a Simulated Annealing Approach." *Assembly Automation* 37 (1): 34-50.

Roshani, A., M. Paolucci, D. Giglio, and F. Tonelli. 2021. "A Hybrid Adaptive Variable Neighbourhood Search Approach for Multi-sided Assembly Line Balancing Problem to Minimise the Cycle Time." *International Journal of Production Research* 59 (12): 3696-3721.

Roy, D., A. Gupta, and R. De Koster. 2015. "A Non-Linear Traffic Flow-Based Queuing Model to Estimate Container Terminal Throughput with AGVs." *International Journal of Production Research* 54 (2): 472–493.

Şahin, M., and Kellegöz, T. 2019. "Balancing Multi-manned Assembly Lines with Walking Workers: Problem Definition, Mathematical Formulation, and an Electromagnetic Field Optimisation Algorithm." *International Journal of Production Research* 57(20): 6487-6505.

Sali, M., and E. Sahin. 2016. "Line Feeding Optimization for Just in Time Assembly Lines: An Application to the Automotive Industry." *International Journal of Production Economics* 174: 54–67.

Sali, M., E. Sahin, and A. Patchong. 2015. "An Empirical Assessment of the Performances of Three Line Feeding Modes Used in the Automotive Sector: Line Stocking vs. Kitting vs. Sequencing." *International Journal of Production Research* 53 (5): 1439–1459.

Saliby, E. 1980. "A Reappraisal of Some Simulation Fundamentals." PhD diss. *University of Lancaster.*

Saliby, E. 1990. "Descriptive Sampling: a Better Approach to Monte Carlo Simulation." *Journal of the Operational Research Society* 41 (12): 1133-1142.

Saliby, E., and R. J. Paul. 1993. "Implementing Descriptive Sampling in Three-phase Discrete Event Simulation Models." *Journal of the Operational Research Society* 44 (2): 147-160.

Salveson, M. E. 1955. "The Assembly Line Balancing Problem." *The Journal of Industrial Engineering* 6: 18-25.

Schmid, N. A., and V. Limére. 2019. "A Classification of Tactical Assembly Line Feeding Problems." *International Journal of Production Research* 57 (24): 7586–7609.

Schmid, N. A., V. Limère, and B. Raa. 2021. "Mixed model assembly line feeding with discrete location assignments and variable station space." *Omega* 102 (2021): 102286.

Scholl, A., and R. Klein. 1997. "SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing." *INFORMS Journal on Computing* 9 (4): 319–34.

Sebban, M., R. Nock, J. Chauchat and R. Rakotomalala. 2000. "Impact of Learning Set Quality and Size on Decision Tree Performances." *International Journal of Computers, Systems and Signal* 1: 85-105.

Setayesh, A., E. H. Grosse, C. H. Glock, and W. Patrick Neumann. 2022. "Determining the source of human-system errors in manual order picking with respect to human factors." *International Journal of Production Research* 60 (20): 6350-6372.

Shaw, P. 1998. "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems." *Principles and Practice of Constraint Programming — CP98*: 417–431. Berlin: Springer.

Simchi-Levi, D., and E. Simchi-Levi. 2020. "Building Resilient Supply Chains Won't Be Easy." *Harvard Business Review.* https://hbr.org/2020/06/building-resilient-supply-chains-wont-be-easy

Sra, S., S. Nowozin, and S. J. Wright, eds. "Optimization for Machine Learning." *MIT Press*, 2012.

Sternatz, J. 2014. "Enhanced Multi-Hoffmann Heuristic for Efficiently Solving Real-World Assembly Line Balancing Problems in Automotive Industry." *European Journal of Operational Research* 235 (3): 740–54.

Sternatz, J. 2015. "The Joint Line Balancing and Material Supply Problem." *International Journal of Production Economics* 159: 304–18.

Tibshirani, R., T. Hastie, B. Narasimhan, and G. Chu. 2002. "Diagnosis of Multiple Cancer Types by Shrunken Centroids of Gene Expression." *Proceedings of the National Academy of Sciences* 99 (10): 6567–6572.

"The backbone of our large-scale production." 2022. *AUDI Brussels* https://www.audibrussels.be/brussels/web/en/production/departments/assembly.html

Thomopoulos, N. T. 1970. "Mixed Model Line Balancing with Smoothed Station Assignments." *Management Science* 16 (9): 593–603.

Thürer, M., N. O. Fernandes, M. Stevenson, T. Qu, and C. D. Li. 2018. "Centralised vs. Decentralised Control Decision in Card-Based Control Systems: Comparing Kanban Systems and COBACABANA." *International Journal of Production Research* 57 (2): 322–37.

Tompkins, J. A., J. A. White, Y. A. Bozer, and J. M. A. Tanchoco. 2010. *Facilities Planning* 110 - 113. John Wiley & Sons.

Van Alstyne, M. W., and G. G. Parker. 2021. "Digital Transformation Changes How Companies Create Value." *Harvard Business Review* https://hbr.org/2021/12/digital-transformation-changes-how-companies-create-value

Verma, S. P., and A. Q. Ruiz. 2006. "Critical Values for Six Dixon Tests for Outliers in Normal Samples up to Sizes 100, and Applications in Science and Engineering." *Revista Mexicana de Ciencias Geológicas* 23 (2): 133-161.

Vilarinho, P. M., and A. S. Simaria. 2002. "A Two-Stage Heuristic Method for Balancing Mixed-Model Assembly Lines with Parallel Workstations." *International Journal of Production Research* 40 (6): 1405–20.

Vitale J., J., and C. Giffi. 2020. "Industry 4.0 in Automotive." *Deloitte Insights* https://www2.deloitte.com/content/dam/insights/us/articles/automotive-news_industry-4-0-in-automotive/DI_Automotive-News-Supplement.pdf

Waters, T. R., V. Putz-Anderson, and A. Garg. 1994. "Applications Manual for the Revised NIOSH Lifting Equation." *National Institute for Occupational Safety and Health, Division of Biomedical and Behavioral Science*, 1–107.

Wee, D., R. Kelly, J. Cattell, and M. Breunig. 2015. "Industry 4.0 How to Navigate Digitalization of the Manufacturing Sector." *McKinsey & Company* `https://www.mckinsey.com/capabilities/operations/our-insights/industry-four-point-o-how-to-navigae-the-digitization-of-the-manufacturing-sector`

Wesselhöft, P. 2020. "Project "P": A Blank Slate at Audi." *Porsche Consulting The Magazine*, September 7. `https://magazine2020.porsche-consulting.com/en/audi-project-p/`

Wijnant, H., N. Schmid, and V. Limére. 2018. "The Influence of Line Balancing on Line Feeding for Mixed-model Assembly Lines." In *32nd annual European Simulation and Modelling Conference 2018*. Ghent, Belgium.

Yin, Q., and X. Luo. 2020. "A Three-Stage Optimization Method for Assembly Line Balancing Problem." *IEEE Access* 8: 143607-143621.

Yilmaz, Ö. F. 2020. "AUGMECON2 Method for a Bi-objective U-Shaped Assembly Line Balancing Problem." In *International Conference on Learning and Intelligent Optimization*: 158-167. Cham: Springer.

Zangaro, F., S. Minner, and D. Battini. 2021. "A Supervised Machine Learning Approach for the Optimisation of the Assembly Line Feeding Mode Selection." *International Journal of Production Research* 59 (16): 4881-4902.

Zangaro, F., S. Minner, and D. Battini. 2022. "The Multi-manned Joint Assembly Line Balancing and Feeding Problem." *International Journal of Production Research*. Advanced online publication. doi: 10.1080/00207543.2022.2103749.

Zennaro, I., F. Serena, M. Calzavara, D. Battini and A. Persona. 2020. "A parts feeding model for big size products: the active and idle assembly islands strategy." *International Journal Advanced Operations Management* 12 (3): 211–236.

Zhang, H., and J. Su. 2008. "Naive Bayes for optimal ranking." *Journal of Experimental & Theoretical Artificial Intelligence* 20 (2): 79-93.

Zhang, H. 2004. "The Optimality of Naive Bayes." In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference FLAIRS 2004* 1 (2): 1-6.

Zhang, M., R. Batta, and R. Nagi. 2009. "Modeling of workflow congestion and optimization of flow routing in a manufacturing/warehouse facility." *Management Science* 55(2): 267-280.