Automating balancing and sequencing of assembly lines in an automotive manufacturing plant

Didden, J.B.H.C.

Award date:
2020

Automating balancing and sequencing of assembly lines in an automotive manufacturing plant

VDL Nedcar

Master Thesis

2019 - 2020

J.B.H.C Didden

DC 2020.031

TU/e Coach: dr.ir A.A.J Lefeber
TU/e Supervisor: Prof. dr. ir. I.J.B.F Adan
Company Supervisor: I. Panhuijzen MSc

Manufacturing Systems Engineering
Department of Mechanical Engineering
Dynamics and Control (D&C) group
Eindhoven University of Technology

Eindhoven, April 2, 2020

Abstract

An important and highly complex process in the automotive industry is the balancing and sequencing of the assembly lines. Optimally distributing jobs among the lines in order to obtain the highest efficiency, and deciding the sequence of the car models in the line is a continuous process, mostly done manually. This thesis aims at automating both processes. Automotive assembly lines are highly complex, and multiple factors have to be considered while balancing the lines. All relevant factors that are found within the lines at VDL Nedcar are considered, namely: Mixed-Model production, sequence dependent setup times and variable workplaces with multiple operators. A MILP model is formulated for both the balancing and sequencing problem, taking all factors into account. The sequencing problem is solved optimally through a branch-and-bound method. A Genetic Algorithm (GA) is proposed to solve the formulated balancing problem. Results show that the proposed GA is effective in balancing a real world assembly line, and, can increase the efficiency of the line compared to the current balance. To the best of our knowledge, this is the first time all these factors have been taken into account simultaneously in a model used for automating the line balancing, and has been tested for a real world case.

Preface

This thesis concludes my final project for the master Mechanical Engineering at the Eindhoven University of Technology. During my six years of being a student I have gained a lot of knowledge and have been presented with amazing opportunities. From doing an exchange semester in Singapore, to doing projects for the only automotive factory in the Netherlands (which I presume countless other students would also love to do), I will always happily look back at my student days.

During this project I had a chance to take a look at automating a key process at VDL Nedcar. I enjoyed every single moment I got to work on this project, and even did not mind getting up early in the morning to travel more than an hour to get to work. Besides my main projects, I also had the opportunity to sit in on countless meetings and help with other smaller projects at VDL. This gave me a great insight in the workings of the entire factory and also helped me with my own project. I would like to thank a few people for all their help and support throughout this project.

Firstly, I would like to thank Erjen Lefeber who coached me during my project. His advice during our weekly meetings has always been of great help. His feedback taught me a lot of how to setup and conduct research, structure and write a report and various other things that will certainly help me during the rest of my career. I am thankful for the opportunity to work together with him.

Next, I would also like to thank Ivo Adan for setting me up at Nedcar. Even though we did not have a lot of meetings, the meetings we did have gave me new insights into the project and always steered me in a new direction that I have not explored before.

Of course I would like to thank all my colleagues at Nedcar. Everytime I had a question everyone was always willing to help immediately. I especially would like to thank my supervisor, Ivo Panhuijzen, for the great guidance during this project. Our weekly monday afternoon meetings helped tremendously with the progression of the project, and without them I would never have gotten the same results. I would also like to thank a fellow intern, Koen, for all the endless discussions we had regarding the project and for all the insights he gave me. Lastly, I would also like to thank Marc and Marijke for all their help.

Last but not least, a special thanks to my parents and brother for all their support (and making sure dinner was ready when I got home) and to my friends and girlfriend for all their support and fun distractions.

Jeroen Didden
Born, April 2, 2020

Contents

List of Figures

List of Acronyms

2ALBP Two Sided Assembly Line Balancing Problem.

ALBP Assembly Line Balancing Problem.

ALX Assembly Line X.

CSP Car Sequencing Problem.

FAS Final Assembly Shop.

GA Genetic Algorithm.

ME Manufacturing Engineer.

MILP Mixed Integer Linear Program.

MMALBP Mixed Model Assembly Line Balancing Problem.

OEM Original Equipment Manufacturer.

PD Process Development.

SALBP Simple Assembly Line Balancing Problem.

CHAPTER 1

Introduction

1.1. Company Background

VDL Nedcar is an independent car assembly factory and the only one in the Netherlands. It has over 50 years of experience in the manufacturing of cars being built for different manufacturers including DAF, Volvo, DaimlerChrysler and Mitsubishi. At the end of 2012 NedCar was acquired by the VDL group who modernized the production-halls and lines. Currently VDL NedCar is assembling cars for the BMW group. Three different models are being produced for BMW: The MINI Countryman (including the PHEV), the MINI convertible and the BMW X1 as can be seen in Figure 1.1.

(a) The MINI Convertible

(b) The MINI Countryman

(c) The BMW X1

Figure 1.1: The different cars produced at VDL NedCar

The factory consists of four different shops: The press shop, body shop, paint shop and the final assembly shop (FAS). In the press shop hundreds of tonnes of steel are processed each day to manufacture all the body parts needed to build the body of the different car models. At the body shop high tech robots assemble the parts that arrive from the press shop into a finished car body that is ready for paint. The body shop is the most automated part of the whole assembly process as almost all processes are conducted by robots. A total of 1300 robots are used in 50 fully automatic production lines. After the body is completely assembled and checked, it moves onto the paint shop. In the paint shop the body is cleaned, degreased, coated with a primer and finally painted to specification. At last the painted body is moved to the FAS. Here, all components of the car are fitted to customer specifications on a 1.7 km long assembly line. After leaving the assembly line the cars go through various checks and inspections before being delivered to the customer.

1.2. Project Background

Comparing all of the different production halls at VDL Nedcar, the Final Assembly Shop (FAS) is one of the least automated. This is due to the complexity of the tasks and the changing demand in different car types, which can highly influence the order in which the tasks have to be conducted. Most of the operations on the assembly line require manual labor, with only a small part (such as installing the front window) being automated.

Finding the optimal order, configuration and amount of workers for all the processes that have to be done to minimize the total production time, is very challenging due to the enormous amount of constraints. In case of constant demand of all vehicle types this problem is fairly simple, as once the line is sufficiently balanced, no major updates have to be done to re-balance the line. In the case of fluctuating demand, as is the case at VDL Nedcar, the line will have to be re-balanced regularly. I.e., if the demand for the Mini Convertible is higher, more time will have to be allocated to installing a convertible roof as this operation will occur more frequently (hence making this process more efficient will save time in the long run). Additionally, an increase or decrease in demand might result in a change in takt time. With this change more or less processes can be allocated to a workstation, therefore rebalancing of the line is necessary. Seeing as the automotive market is highly volatile, and demand changes can occur rapidly, flexibilty is also of importance. Being able to quickly rebalance the lines according to customer demands can therefore help meet these changing demands.

Also, the constant updates of the different models influence the way the line has to be balanced. A small change in car parts can result in different tooling that has to be used, which consequently results in more/less time that has to be allocated to the process. This again can result in another process that has to be changed at a workstation to make sure the total process time of a workstation does not exceed the takt time (in the case that a process requires more time) or that a workstation is not idle for too long (in the case that a process requires less time).

An even bigger change would be when a new vehicle model is introduced into the existing assembly line. As VDL Nedcar operates a mixed model assembly line, producing all models in random order, balancing the line when a new model is introduced takes a vast amount of time, as all processes for all models have to coincide with each other. Even with "perfect" data it would take at least 3 weeks to balance the line. However, in this case the efficiency of the line would be at 70%, while VDL Nedcar desires an efficiency of over 90%. From experience it is known that it would take at an additional least 3-4 weeks to achieve this.

The reason that at the moment manual balancing of the line is done, instead of automated balancing, is due to the high complexity of the problem. A considerable amount of parameters, constraints and goals has to be taken into account which is usually done by manual interpretation. Translating these constraints into a mathematical model that can be solved by a computer is dif-

ficult. However, automating the line balancing can potentially save a tremendous amount of time and money in the long run.

For these reasons, VDL Nedcar is searching for a way to automatically balance the assembly lines, saving a vast amount of time over the long run and to be flexible to quickly meet the changes in customer demand. This allows the current engineers, instead of balancing the lines by hand, to spend more time on improving the processes (i.e., making the process more efficient or improving the ergonomic workload), improving the quality of life of the operators and improving the overall product. Furthermore, by automatically balancing the lines, more feasible solutions can be found for the problem. More choices can be explored in a shorter amount of time and, depending on the method used, an optimal solution may even be found.

1.3. Objective

The overall objective of this project is to develop a model to automatically balance the assembly lines at VDL. As this is a fairly novel and elaborate project, the main objective is split up into multiple subobjectives, all based on the following main research question:

How can the current assembly lines be balanced automatically so that, when changes are made to the assembly line (i.e., process or volume), the highest possible efficiency is achieved?

In order to answer this specific research question multiple subquestions will have to be answered:

- How are the assembly lines currently being balanced? What can be learned from this process?
- Which constraints of the assembly process need to be taken into account? Or do not have to be taken into account? How do these constraints need to be defined?
- What input data is available? What is the desired output? Which extra input data is needed to achieve the desired output?
- How can fluctuations in demand be taken into account?
- Is the line balance always feasible independent of the order of the cars in the assembly line?

1.4. Structure

The report is structured as follows: in Chapter 2 a more elaborate explanation of the problem is given, including a description of the assembly lines and the current way the lines are balanced. Then, in Chapter 3 the balancing model is described with an elaborate explanation of how the various constraints that define the assembly line need to be taken into account. Chapter 4 and 5 give solution methods to solve the problem. Lastly, Chapter 6 explains how the model is implemented at VDL and shows the result and the conclusion and recommendations are given in Chapter 7.

CHAPTER 2

Problem Background

2.1. Assembly Line Balancing

Balancing of an assembly line is one of the most challenging and complex tasks conducted at VDL Nedcar. Incorrectly balancing an assembly line can cause a significant inefficiency in the line, resulting in more capacity being consumed than is necessary and creating a surplus of idle time (i.e., time that is not spent executing a task). Two different types of assembly lines are present at VDL Nedcar, however the type that is most influenced by line balancing is explained and has been used throughout the remainder of this study. An *assembly line* is a large conveyor belt that consists of multiple *workstations*. Workpieces are launched down the line at a predefined time interval, commonly referred to as the *takt time* and continously move along the line. At each workstation specific *tasks* are conducted to assemble the product. These tasks are done within the operators boundaries and are then repeated when a new workpiece enters the workstation (i.e., tasks are repeated in a cyclic sequence). The tasks are carried out by *operators* assigned to certain *mounting positions* of the workpiece at every workstation. Considering the size of the workpiece (a typical car is around 4500 mm long), multiple operators can be assigned to a workstation. The problem of optimally assigning all the available tasks to the available workstations is known as the *Assembly Line Balancing Problem (ALBP)*, see e.g. Baybars (1986). See figures 2.1 and 2.2 for schematics of the assembly line and a single workstation.

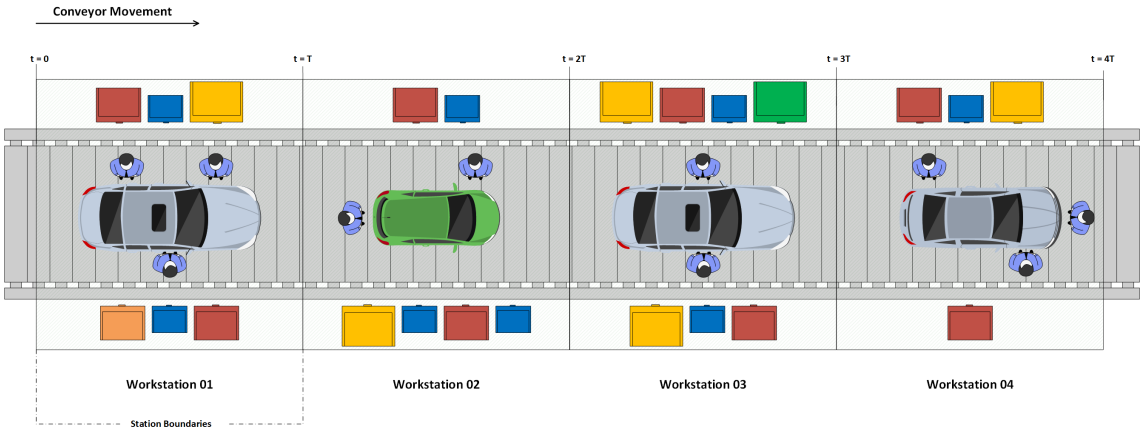


Figure 2.1: Schematic overview of a piece of the assembly line. Different consecutive workstations are shown including its boundaries and length in terms of the takt time.

2.2. Assembly Line Description

The main focus for this project is on the FAS, therefore this is explained in slightly more detail. The production lines have a total length of 1.7 km and contain about 400 workstations total. There is one main assembly line and several sub-assembly lines which are used to assemble various (sub-)parts of the car. The main assembly lines consist of 14 line sections which each have between 5 and 25 workstations. At the end of each section the car body is transported via an overhead transportation system to the next line section. This transportation system can function as a buffer between the sections. The buffer consists of a track and hanger. At the beginning and end of every line a lift ensures that the car bodies can be transported above other lines to make efficient use of the floor space. In addition, as the bodies are transported with the use of hangers, a return loop of empty hangers is required in the system. The length of the overhead transportation system varies between each line. This buffer system is also a way to decouple the different assembly lines from each other, making it possible, although it is not required, to balance all 14 line sections separately.

Multiple factors have to be accounted for when balancing the assembly line. Each of these factors influence the way the line can be balanced in different ways. Some factors influence the balancing more than others. The most important factors to consider and the factors that cause the greatest disturbances in the line are Mixed-Model, Paced Line, Zoning and Tooling, Ergonomics, Setup Times and Large tasks and these are explained below.

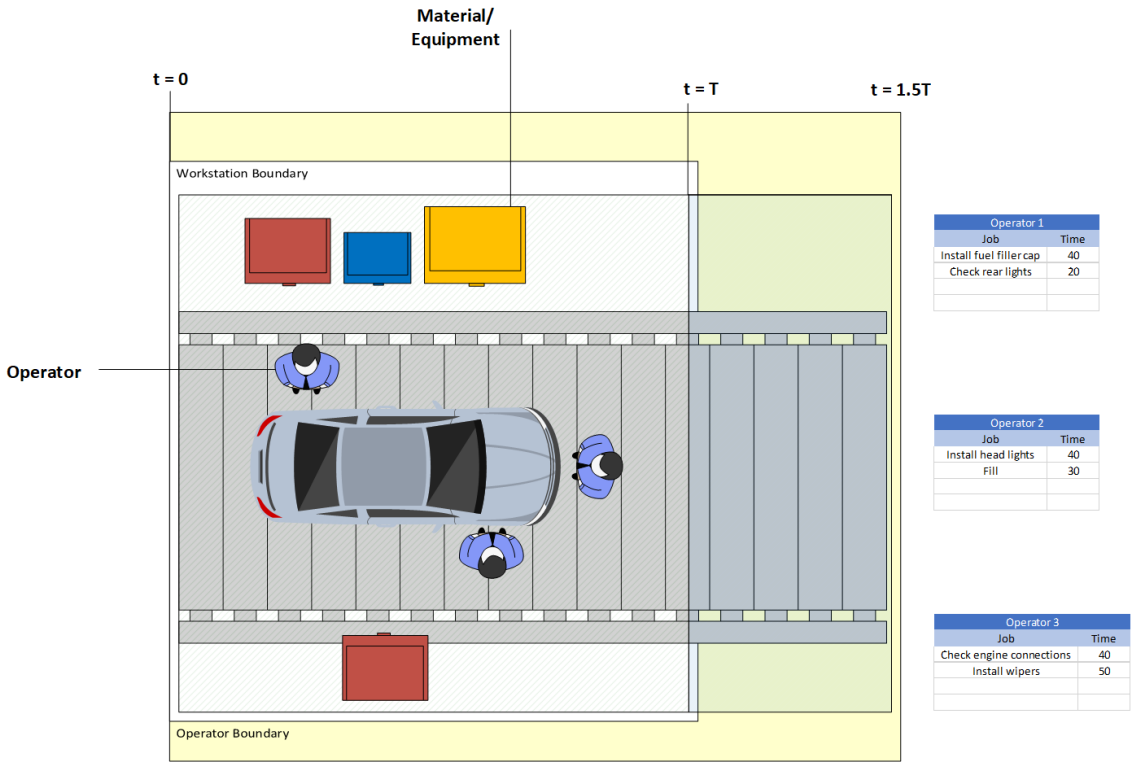


Figure 2.2: Detailed schematic of a single workstation including an example of various tasks and processing times allocated to each operator on the workstation. The operators boundaries are not the same as the workstation boundaries as an operator can exceed the takt time occasionally. The tasks allocated to a worker can be unique for a single model

2.2.1. Mixed-Model

In total three different (main) models are produced on the assembly line: The Mini Convertible, the Mini Countryman and the BMW X1. All three of these models are produced in a make-to-order fashion, meaning there is a random order (according to some predefined rules) in which the models are produced (no batch production). Furthermore, all models can have different options as desired by the customer. E.g., while one customer may want parking sensors, another customer might not.

Consecutive cars cannot overtake each other in the line, even if a car has less overall processes than the preceding one. Every car has to pass every workstation on each assembly line, even though it possibly does not have tasks at every workstation. This causes a high level of variation of workload between cars in the assembly line. The order of cars is not randomly chosen, they adhere to specific rules laid out by the planning department. Due to the way the line is balanced, some specific models cannot be planned too close after each other (e.g., there must be a certain gap between two similar models as otherwise the cycle time might be exceeded too much which causes problems throughout the entire line). The order in which the cars enter the assembly line is referred to as the car entry sequence.

Typically, not only the main models are sequenced throughout the line. Some sub-models are more complex with regards to the main model (e.g., a plug-in hybrid) and some options require a large amount of time to be installed (e.g., a sunroof). These sub-options are therefore planned separately alongside the main models. Typically, planning is done on a priority based system, so that the complex variants can be planned first.

2.2.2. Paced Line

All assembly lines act according to a fixed, predefined takt time. This takt time determines the pace of the line, the total time at which a car body is present at each workstation. The takt time is given in cmins ($\frac{1}{100}^{th}$ of a minute). The objective of balancing is to pre-allocate each process, so that the total time of each process at a workstation does not exceed the takt time for the assembly line. Optimally, the total time should also not be less than the takt time, as this might lead to non-useful idle time. Lastly, the car bodies move with a continuous velocity through the assembly line and are never stopped.

2.2.3. Zoning and Tooling

Zoning can occur in two ways on the assembly line: either referring to the position where the operator is working on the car or limitations of the positioning of the tool. The latter is referred to as tooling restrictions.

The assembly line itself consists of a main conveyor which transports the car bodies between the workstations. Depending on the part of the assembly line, operators can work at different zones of the car. In some cases operators work at both the left and right side, in other cases the operators can work at four sides of the car, as long as the processes do not coincide with each others work spaces. Each operator belongs to a single workstation and a group of workstations belongs to a group. A group is defined as multiple operators who work on the line simultaneously on different zones.

The tools that need to perform certain processes also have zoning restrictions. Some processes can only be done at certain workstations as the tool necessary to perform the task is only available at that workstation. Secondly, there must also be enough room at a workstation to perform certain tasks. For larger workpieces there must be enough room for the operator to maneuver and install the workpiece. Lastly, some processes have certain height restrictions. E.g., some tasks have to be

performed at the top side of the car, therefore the task has to be planned on a workstation where the operator can easily access the top of the car.

2.2.4. Ergonomics

The ergonomic workload of the tasks is one of the most important factors. This typically refers to the physical load of a specific task, and in lesser extent, to the psychological effect repetitive tasks can have on a operator. Even though the weight of a physical task can differ between operators, the workers union defines what the ergonomic load for a specific task is (based on multiple factors). The aim when balancing the line is to keep the total ergonomic weight of all tasks for a operator as low as possible and also below a certain threshold. Different methods can be chosen to balance the ergonomic workload for an operator, however taking this into account while automatically balancing the line is difficult, as some tasks exceed the threshold. To counteract this workers should be able to switch workstations during their shift in order to keep the overall workload to a minimum. As this is a difficult factor to model (i.e., for now it can more cleverly be done "by hand"), it will remain out of the scope of this project.

2.2.5. Setup Times

The factor that influences the line balancing the most is setup times. Setup times are created when an operator has to walk from one mounting position to another mounting position (and on the way grab new material/equipment). Due to the relatively short takt time, large setup times between different tasks can greatly influence the amount of operators needed in the line (i.e., the setup times can consist of more than 10% of the workload of a workstation). The setup times are also sequence dependent, as the order in which the tasks are sequenced on a workstation determines the amount of setup time accumulated on the same workstation.

2.2.6. Large tasks

Typically the duration of a task is less than the takt time, however due to certain circumstances this may not always be the case. Some smaller tasks are combined to form a larger task that exceeds the takt time (sometimes by more than double the takt time). This is done for either quality reasons or to more evenly distribute the ergonomic workload on an operator (i.e., so that physically demanding tasks are not carried out by a single operator every cycle but rather every n cycles).

In addition some tasks also require the use of multiple operators (e.g., installing the headliner). In this case two or more operators simultaneously execute the same task, but on different sides of the car. When planning these tasks, it must therefore be taking into account that the tasks start at the same time even though they are executed on different sides.

2.3. Assembly Line Sequencing

The secondary problem that is considered is the sequencing of the assembly line. Sequencing refers to the production planning of the models (i.e., the order in which models and its variants are launched down the line). Mixed-Model lines are often balanced on average demand, causing some models to exceed the takt time, which are in turn compensated by models that have a station load less than the takt time. Planning too many models with high station times consecutively can cause high amounts of work overload which have to be compensated with either line stoppages or with utility workers. Line stoppages cause a decrease in throughput which can be very costly if it occurs too often and employing utility workers too often can cause other discrepancies in the line (i.e., as the group coordinator is often used as utility worker).

Sequencing is currently done at VDL Nedcar using the Car Sequencing (CS) method. Rules are created considering some options of the available model variants. The rules are typically based on the amount that the takt time is exceeded, and the amount of time that can be compensated with

variants that are lower than the takt time (e.g., if the options sunroof takes a considerable amount of time, then it can be compensated with cars that do not contain a sunroof). The maximum distance that an operator is allowed to digress (i.e., the operators boundary) from his assigned workstation is then used to calculate how many times a variant can be sequenced consecutively. Currently, there are no strict equations that are being followed to calculate the rules, most rules are made by an educated guess, however the aforementioned variables are still considered.

Not every rule can always be adhered to completely, as this depends on the amount of rules created and the demand for each variant. The assembly line is then sequenced according to all generated rules by minimizing the amount of violations caused by every rule. It must be noted that not every variant obtains a specific rule, typically only the variants that cause a large amount of overload are considered while planning, since there can be thousands of unique cars. Usually models are sequenced on the options they contain (e.g., sunroof).

2.4. Balancing at Nedcar

The engineers at VDL typically follow the same basic scheme for balancing with only some slight variations, e.g., due to their experience.

The required tasks that need to be executed to completely assemble each model are delivered to VDL by the OEM, while more specific information such as the required task duration is not delivered by the OEM. All these tasks require a certain time frame in which they expect a fit and healthy person to be able to conduct the process. The time frame is calculate by the Process Development Engineers (PD-engineers) through a Methods-Time Measurement (MTM-2). The task durations are calculated according to regulations in the workplace, but are always double checked on the line themselves as the times can differ greatly in some cases. In addition the variance in the task durations is also checked as variances can cause uneven loading of a workstation.

After information is obtained from the customers, the PD Engineers adapt this information to comply with the lines at VDL. The PD engineers supply more information such as the process number, description, equipment needed, task duration and model variant. For clarity, a difference has to be made between a process and a task. A process is a group of tasks, e.g., the process of installing the muffler consists of multiple tasks such a placing screws.

The information supplied by the Process Development (PD) engineers is checked by the Manufacturing Engineers (ME) for possible discrepancies. After this, the balancing engineers sequence the processes on the workstations available. The engineers make their own decisions based on the description of the process, process time, tooling, zoning and experience. This is done by manually 'dragging' the tasks to feasible workstations. The total workload is checked against the takt time. The constraints are applied manually (such as precedence, zoning, workload) by the ME, where improvements are done by a trial-and-error approach and aided by feedback. The precedence constraints are not specified in the information provided by the OEM. Typically, the task will be distributed over the workstations in a ascending order according to their task number (in other words, it is assumed that task $n+1$ is a direct predecessor of task n, while this does not have to be the case). The processes however can be distributed in a different order than the precedence relations. tasks are grouped together in processes, as some tasks have to be done consecutively (e.g., grabbing and installing the same part). Lastly, the engineer decreases walking times by assigning tasks in specific orders. tasks are scheduled so that an operator walks the least amount of distance in between new tasks.

On average the engineers spend about a quarter of their time balancing the lines, sometimes more depending on the amount of updates that need to be processed. There are four main reasons why the assembly line would need re-balancing (other reasons also exist, but they are less frequent or influence the balancing to a much smaller extent):

1. Model Update: This occurs when a current model is updated by the OEM (usually once a year). Due to the slightly different design the models can have, the processes needed to assemble the car can change. This will affect the entire assembly line

2. Process Update: This occurs when a change is made to an existing process in the line. A more efficient method to a task has been found or a different way to mount a certain workpiece. This usually only effects a single line section initially.

3. Model introduction: Although this is a unique process that only occurs sporadically, it requires substantial re-balancing of the entire line for the newly introduced model. The current models on the line will not have to be rebalanced completely (i.e, the new model will be made to fit on the current line).

4. Demand changes: The total demand may change, which causes a significant change to the takt time of the line. The line will have to be rebalanced to account for this. Smaller demand changes regarding specific model also occur, this requires less dramatic rebalancing.

The process itself is continuous, meaning that an update for the line can come at any given moment. The engineers do not describe the process of line balancing as difficult, only that their main challenge is the amount of processes that each of them have to keep track of (more than 100 per engineer). Each engineer may also be biased in the way they balance their lines and finally, due to the amount of time a single balance takes, only a handful of solutions can be tested at any given time.

2.5. Literature Review

A lot of review papers have been written for the Assembly Line Balancing Problem that classify all the current literature. Multiple papers (see for example Battaïa and Dolgui (2013), Becker and Scholl (2006), Boysen et al. (2007), Li et al. (2017), Tasan and Tunali (2008), Boysen et al. (2008) and Sivasankaran and Shahabudeen (2014)) make comprehensive reviews of all the research that has been done in the past decades. All these papers look at the different options that are available in the ALBP (e.g., paced or unpaced, manual or automated etc.) as well as the various algorithms that are available to solve such problems. They divide the solvers into 3 main groups; mathematical optimization, exact solution procedures and meta-heuristics. All of the aforementioned review papers come to a similar conclusion; even though there is enough literature describing all of the problems concerning the ALBP, only a small part of it covers actual real world problems. Most of the listed papers in the review articles cover enough aspects of the ALBP, yet a very large portion have no relevance towards real-world problems (i.e, the size of the problems is not comparable). Boysen et al. (2008) describes that of the 312 papers treating the ALBP, only 15 cover real world assembly lines and two of those cover automobile production. It therefore becomes unclear whether the proposed solution methods in the current research are applicable to the real-world problems such as these at VDL. In addition, most of the academic research does not represent the re-balancing of the line but rather first time installation. The difference here is that current lines often have fixed places for heavy machinery which is costly to move, see Boysen et al. (2008). Also, some operators may have been trained to conduct certain tasks in a fixed order, which may result in training costs when the order of the tasks is changed. This causes some design variables to be fixed and limits the amount of options.

Boysen et al. (2008) furthermore propose several aspects of assembly line balancing in the automobile branch that can/have to be taken into account. These are: mixed model production, parallel working places within a workstation (multiple operators working on the same workpiece), parallel stations (typically referred to as the Two-Sided Assembly Line Balancing Problem (TALBP)), unproductive activities at a station (i.e., operator switching station). Lastly, an important function to minimize is that of the so called horizontal balancing ,i.e., minimizing the variance in the station

times for all model types.

Falkenauer (2005) furthermore emphasizes the problems that are also seen in the works of Becker and Scholl (2006). Even though a lot of literature is available solving both ALBP, only a few constraints are taken into account while real world practices deal with multiple constraints. Falkenauer (2005) argue that, especially within the automobile industry, constraints such as the re-balancing of the line, zoning, ergonomics and other operator constraints must all be taken into account simultaneously. This in turn gives rise to a severely complex problem. The SALBP, and therefore also the GALBP and MALBP, are typical NP-hard problems (as discussed by Álvarez-Miranda and Pereira (2019) and Wee and Magazine (1982)) or Bin-Packing problems (as discussed by Wee and Magazine (1982) and Álvarez-Miranda and Pereira (2019)).

More recently, Alghazi and Kurz (2018) adapted the SALBP and took zoning, ergonomic workloads, parallel workstations, mixed-model assembly and assignment restrictions into account. They used Integer Programming (IP) and Constraint Programming (CP) to optimally solve the problem (by using the least amount of workers). They showed that for big problems (using information provided by an independent OEM) CP models outperform IP models.

It becomes clear that there is a big gap between the academic research done on ALB and real world examples. In addition, the academic research mainly focused on the SALBP and the GALBP where only a handful of additional constraints have been tested, usually separate from each other. A lot of different solution methods have been tested over the past decades, possible making it simpler to find a suitable method that can be implemented at VDL. It can be concluded that a lot of care has to be taken when applying some of the aforementioned models to the assembly line at VDL and a lot of attention has to be payed to which constraints have to be applied.

2.6. Summary

In this chapter an explanation of the assembly lines at VDL Nedcar was given. The assembly line consists of multiple workstations, on which multiple operators can work. To each operator tasks are allocated, which they have to complete within the lines takt time. Next, the most important factors that need to be considered while balancing the line were described. This includes of Mixed-Models, paced lines, mounting positions, tooling and setup times. Lastly, an explanation was given of how the lines are currently being balanced and how the current data is built up.

In the next chapter an elaborate explanation of how to incorporate the various factors that influence the line balancing into a solvable problem is described. Later, in chapters 4 and 5 solution methods are given to solve the problem.

CHAPTER 3

Model Description

In this chapter a description of the proposed model is given. First the most basic assembly line balancing problem is described which forms a base for later additions. After this the additions to the model are given based on the assembly line description as given in Chapter 2. The most important and complex additions to the simple model are sequence dependent setup times and mixed model lines. Both of these additions are described in more detail, next to some smaller additions.

3.1. Assembly Line Balancing Problem

In this section the theory needed to solve the ALBP is given, coinciding with the description of the assembly line as given in Chapter 2. First, the basis of the ALBP, referred to as the Simple Assembly Line Balancing Problem is given. After this, extensions to the model are provided based of relevant literature and information from experts at Nedcar.

3.1.1. Simple Assembly Line Balancing Problem

The most basic form of the assembly line balancing problem is referred to as the Simple Assembly Line Balancing Problem (SALBP), by Salveson (1955) and Baybars (1986). The bin packing problem serves as a base for the SALBP with the addition of precedence relations.
Every assembly line has a set of tasks which must be done, so that the products that cycle through the assembly line are made to requirement (see Baybars (1986) and Boysen et al. (2007)). A task here is defined as the simplest action that can be performed so that it cannot be separated into smaller tasks. The tasks needed to complete the workpiece can be characterized by the set $j \in \{1, .., J\}$. For each of the tasks j a processing time τ_j is given, which in turn describes the length of time that can be spent on a process. Along the assembly line are a number of workstations $w \in \{1, ..., W\}$ the tasks must be divided over. Each line has a predefined takt time $\mathcal{T}$, the time at which a new model is introduced into the line. Due to this, all the workstations have a maximum processing time equal to the takt time. The sum of all tasks done at every workstation must therefore be at most equal to the takt time. There are multiple objectives for the SALBP:

SALBP-1 Given a fixed takt time, the objective is to minimize the number of workstation used.

SALBP-2 Given a fixed amount of workstations, the objective is to minimize the takt time, thus maximizing the output.

SALBP-E Altering both takt time and number of workstations, the objective is to maximize the efficiency of the assembly line. The efficiency is defined as the fraction of the total time used, t_{tot}, and the total available time, $W \cdot \mathcal{T}$.

SALBP-F Given a fixed number of stations and takt time, the objective is to find a feasible line balance.

Furthermore, the tasks must comply with precedence relations, which describe the relationship between the different tasks corresponding with the order in which the tasks must be executed and their physical limitations. These precedence relations can be represented in a precedence diagram, where the nodes are the tasks and an arc between nodes i and j represents a precedence relation (with i being an immediate predecessor of task j). An example of a precedence diagram is given in Figure 3.1. A feasible line balance is achieved when no precedence relation is violated and when no task exceeds the takt time. In case that the last assigned task is completed before the takt time then the remaining time is unused.

In the SALBP the following assumptions are made to relax the problem (see Baybars (1986), Becker and Scholl (2006) and Boysen et al. (2007)):

Assumption 1 The task times are all deterministic and no variations in duration are considered.

Assumption 2 All tasks must comply with their precedence relations.

Assumption 3 Tasks are launched down the line at a predetermined time interval, i.e., the takt time is fixed and does not vary.

Assumption 4 Feeder or parallel lines are not considered.

Assumption 5 A task can only be processed on a single workstation and cannot be split up between workstations.

Assumption 6 Only a single model is produced on the assembly line and this model does not have any variants.

Assumption 7 Every task can be processed on every workstation, i.e., there are no equipment or material restrictions present.

Assumption 8 Only precedence relations can cause assignment restrictions.

The first 3 assumptions accurately describe a real-world assembly line, however, all other assumptions simplify the problem too much and can lead to infeasible solutions to the ALBP. As has been described in Chapter 2, an actual assembly line produces multiple models and multiple variants of these models simultaneously. Furthermore, restrictions as the zoning of tasks, setup times, equipment and material constraints should all be considered to come to a usable and feasible solution. Lastly, some tasks may exceed the takt time when this is necessary for the mounting of a large part. For the remainder of this thesis assumptions 1 to 3 are therefore be kept. In addition, feeder and parallel lines are out of the scope of the project. Typically, feeder lines are not the bottleneck in a production line and parallel lines do not exist at VDL. However, assumptions 5-8 are changed into the following:

Assumption 5 Tasks that exceed the takt time can be processed on consecutive workstations and should be assigned to the same operator (i.e., the operator finishes the task before returning to his assigned workstation).

Assumption 6 Multiple models and variations of models can be produced on the same line. The order in which these models are produced are subject to change depending on their demand. Models are not produced in batches, but can be processed at any given moment within the production sequence.

Assumption 7 Workstations have a unique identity depending on the available equipment and space restrictions.

Assumption 8 Tasks can be restricted to certain workstations if special equipment (or material) is needed.

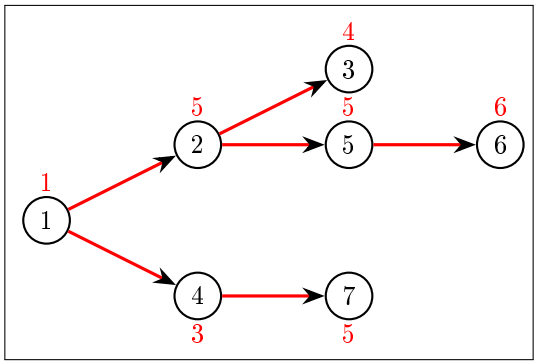


Figure 3.1: Example of a precedence diagram where the black number represent the task number and the red number corresponds to the processing time of the respective task

3.1.2. Sequence Dependent Setup Times

Setup times are a very relevant addition to the SALBP, especially in the automotive industry. However, most researchers do not take setup times into account explicitly (e.g., due to the advances in manufacturing technology quick setup times can be achieved, thus making setup times irrelevant). However, in the automotive industry the relatively large size of a car body and the short takt time can cause the setup times accounting for, on average, 15% of the total station time. Setup time can be caused by a variety of situations. The most applicable ones in the automotive industry are listed below:

Walking distance The length of a typical car is between 4-5 meters. When two tasks are planned successively, but on opposite side of a car body, then the total walking distance can be around 10 meters. If it is taken into account that the average walking speed of a person is around 1 m/s, then this accounts for 10 seconds of walking time. When the takt time is around 60 seconds (which is typical for a car manufacturer), then the walking time has already accounted for 17% of the total station time.

Material changes Another important aspect in setup times is the collection of materials. When switching tasks the operator may have to walk to a material bin to grab new material for the subsequent tasks. The material bins are usually placed alongside the line, at a distance of 1-2 meters depending on the available space. The shortest distance an operator can travel is around 3 meters then, still accounting for about 5% of the station time. However, as operators always carry out multiple tasks within a station, this number increases with the amount of tasks planned.

Equipment changes Similar to the aforementioned material changes, an operator may also have to change equipment between tasks. This again account for a similar amount of setup time as before.

It also becomes clear that the setup times greatly depend on the sequence in which the tasks are planned to a workstation. The objective is to create as little setup times as possible in a workstation, as setup times have no added value. Therefore, it is desired to calculate the setup times between every pair of feasible tasks so that the tasks can be sequenced accordingly in their workstations. A common way of applying setup times is by denoting that if the setup time between a pair of tasks is too large then these tasks cannot be placed on the same workstation, see e.g., Becker and Scholl (2009) and Alghazi and Kurz (2018). This, however, leads to a reduction in scheduling freedom, as feasible line balances remain unconsidered.

Andrés et al. (2006) was one of the first to take sequence dependent setup times into account. They made the considerations that when task j_2 is preceded by task j_1 in a station load, then a

setup time π_{j_1,j_2} must be added to compute the station time. Furthermore, if a task j_2 is the last task in a station load and task j_1 the first task, then a setup time equal to π_{j_2,j_1} must also be added to the station time as the tasks are repeated in a cyclic sequence. The length of the setup times depend on the factors as described previously.

However, as explained by Scholl et al. (2013), a difference has to be made between forward and backward setup times, as these setup times can vary significantly. A forward setup time is caused when a task j_2 is directly preceded by a task j_1 during the same production cycle p (the definition of a cycle is when an operator is processing tasks on the same workpiece). A forward setup time between tasks j_1 and j_2 is denoted as π_{j_1,j_2}. In contrast, a backward setup time is present when a task j_1 is processed directly by the same operator after task j_2 in the next production cycle $p+1$ (i.e., on the next workpiece). A backward setup time must be finished within the takt time of cycle p in order to start task j_1 when cycle $p+1$ begins. A backward setup time between tasks j_2 and j_1 is denoted as μ_{j_2,j_1}. It must also be noted here that a forward setup time between the same tasks cannot exist (i.e., π_{j_1,j_1}), yet on the other hand a backward setup time between the same tasks can exist (i.e., μ_{j_1,j_1}).

Figure 3.2 shows how setups are defined and the differences between forward and backward setups. The simplest setup time is created solely by walking distances, as is the case between tasks j_1 and j_2. Here the operator does not have to change equipment or grab new materials. In contrast, the setup between tasks j_2 and j_3 is more elaborate. The operator must walk from position (a), to the material box at position (b), and then continue to task j_3 at position (c). This creates extra setup time in comparison with the situation that the operator walks directly from task j_1 at position (a) to task j_3 at (c). Next, the difference between a forward and backward setup becomes apparent. If the operator finish cycle a with task j_4 and starts the next cycle $a+1$ with task j_1, then only a short setup time is needed (i.e., the backward setup μ_{j_4,j_1}). However, if task j_4 is succeeded by task j_1 in the same cycle a then a much longer setup time is relevant, namely the forward setup π_{j_4,j_1}.

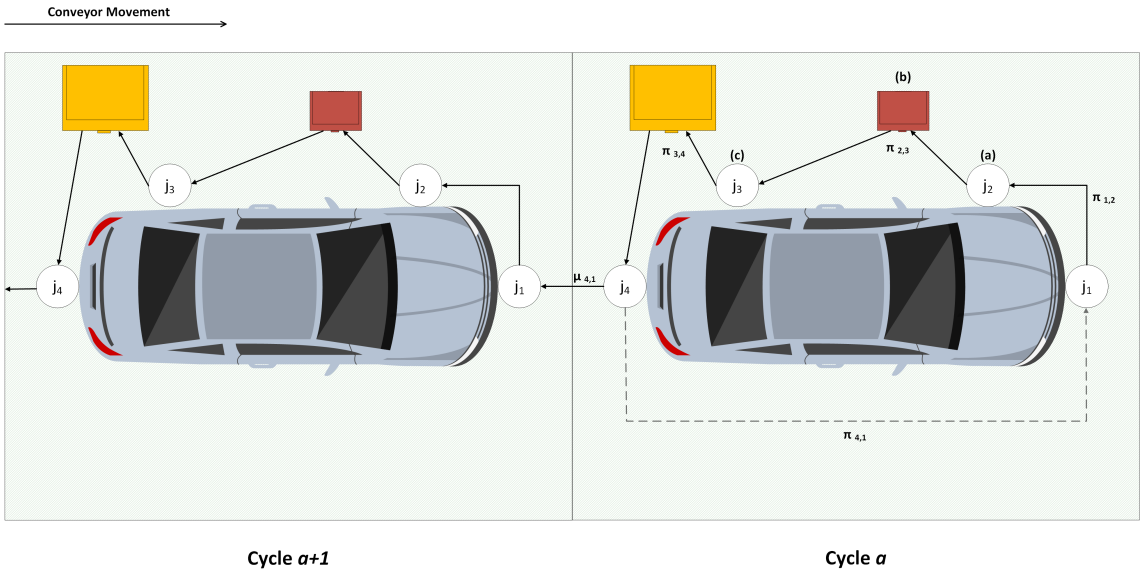


Figure 3.2: The connection between different tasks executed on a workpiece in consecutive cycles

Example 1: Sequencing tasks *Consider the processing time and setup time as given in Table 3.1 if the tasks are ordered in the sequence* $\{1,2,3,4\}$ *then the total station time is equal to*

$\tau_{j_1} + \pi_{j_1,j_2} + \tau_{j_2} + \pi_{j_2,j_3} + \tau_{j_3} + \pi_{j_3,j_4} + \tau_{j_4} + \mu_{j_4,j_1} = 16$. *This is a feasible (and in this case optimal) sequence for the problem. However, consider the sequence* $\{1, 3, 2, 4\}$. *In this case the total station time will be equal to* $\tau_{j_1} + \pi_{j_1,j_3} + \tau_{j_3} + \pi_{j_3,j_2} + \tau_{j_2} + \pi_{j_2,j_4} + \tau_{j_4} + \mu_{j_4,j_1} = 19$, *which is not a feasible sequence for a single workstation. If the tasks are planned in this sequence, then task 4 has to be sequenced on the next workstation.*

This example therefore shows the importance of sequencing the tasks in the workstations correctly. The following assumptions are therefore added to the model:

Assumption 9 Forward and backward setup times are taken into account and are unique to each processed model and model pair.

Table 3.1: Processing times and setup time for the example in Figure 3.2

Description	τ_{j_1}	τ_{j_2}	τ_{j_3}	τ_{j_4}	π_{j_1,j_2}	π_{j_2,j_3}	π_{j_3,j_4}	μ_{j_4,j_1}	π_{j_1,j_3}	π_{j_3,j_2}	π_{j_2,j_4}
Time	2	4	3	1	1	2	2	1	3	2	3

3.1.3. Multiple Operators and Mounting Positions

Balancing the line by only utilizing a single operator is cost effective, however it results in long assembly lines due to the amount of tasks that have to be carried out. The main solution to this problem is by allowing multiple operators to work on the car simultaneously, thus minimizing the length of the line. The size of the car body allows this as operators can work on opposite sides and therefore not interfere with each others work. The car body is split up into different zones and depending on the location of specific tasks, a car can have between 1 and 24 zones. The most simple method of utilizing multiple sides is by considering two zones, being the left and right side of the car. This method is referred to as the 2-sided assembly line balancing problem (2ALBP), see, e.g., Kim et al. (2000). Here tasks are designated with zone information, either containing an L when the task has to be done on the left side of the car, R for the right side and E when it can be done on either side. Different balancing methods have been derived for the 2ALBP, which mainly focus on different solution methods and in turn make decisions on where to plan the E tasks as this can greatly influence the balance.

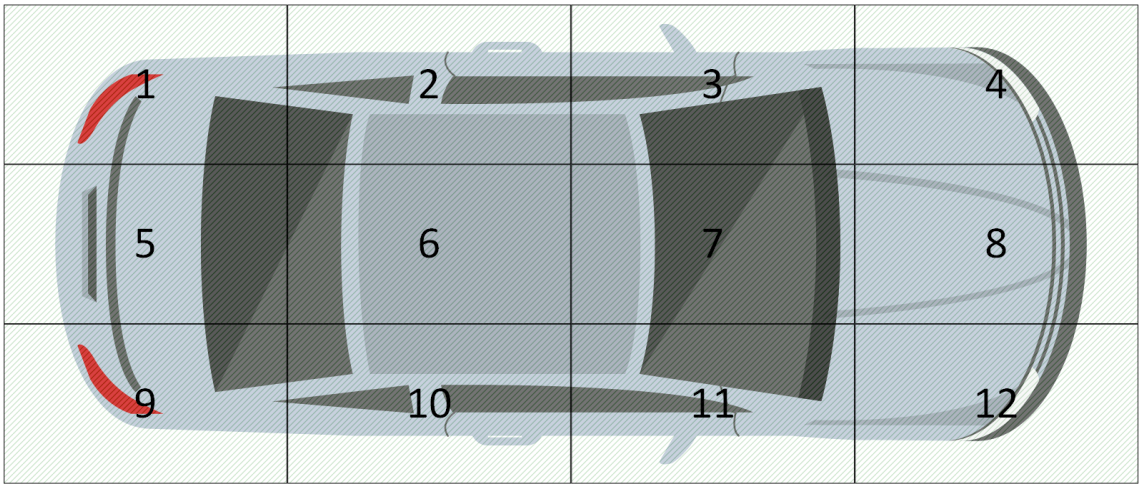


Figure 3.3: An example of a car body divided into 12 different mounting positions. Every operator on the line can take up 1 or more unique mounting position in accordance with with the tasks allocated to the operator

However, the problem by only considering two separate zones is that operators may interfere with one another. If two operators work on tasks that are labeled 'E' at the same time, then there is a possibility that they are both occupying a similar smaller zone (a task inside the car body can be executed on either side of the line, which can lead to interference). The two sided balance does not contain enough information to correctly assign tasks to the operators at every workstation. The 2ALBP can also be extended to an N-ALBP considering N different sides of a workstation. Additionally, these problems typically assign mounting positions to operators and therefore limit the amount of tasks a single operator can execute.

The solution to this is to not fix an operator solely to a single side of the line. All tasks contain specific zone information, which in turn means that assigning tasks to operators also assigns zones to operators. Thus, during the balancing procedures, when a task is assigned to an operator, a mounting position is also assigned to the worker. Other operators can therefore not be assigned to tasks that take up the same mounting position. This idea was first proposed by Becker and Scholl (2009) and is referred to as variable workplaces. Every operator can be assigned every available mounting position depending on the assigned tasks. This allows for more flexibility in the ALBP. In conclusion, the following assumption must always hold:

Assumption 10 Every operator on the assembly can be assigned to any available mounting position, however operators on the same workstation cannot share a zone (i.e., if a task is assigned to an operator then another operator on the same workstation cannot be assigned tasks that occupy the same zone).

Furthermore, when considering multiple operators occupying the same workstation additional constraints regarding the precedence relations must be taken into account. These precedence relations can cause idle times when one operator has to wait for another operator to finish his/her task. Therefore:

Assumption 11 If task j_1 is assigned to a workstation w and task j_2 is a successor of task j_1 that is assigned to the same workstation, then j_2 can only start when j_1 has been completed, independent of the operators assigned to the tasks.

3.1.4. Mixed Model Assembly Line Balancing

In assembly line balancing there are three cases to consider when accounting for different model mixes, see Figure 3.4. The easiest one is single model, where only a single product is made on an assembly line. However this is becoming less common as customers often desire a customized product outfitted according to their needs. Mass production of highly configurable products has become a standard practice, though in the early days this was not the case, see Hounshell (1984). The Ford Model T was the first mass produced product on the market. In 1909 at the Highland Park Complex a Model T was produced every 3 minutes, with the production eventually totaling 2 million vehicles in the year 1925. Over the years the production system became more efficient, significantly dropping the prices of the vehicles. The only difference from today was the customization of the Model T. Every model came with the same engine, wheels, body and color, with the famous quote from Henry Ford: *"Any customer can have a car painted any color that he wants so long as it is black"* Ford and Crowther (1922).

In today's vehicle market the amount of customization is enormous, with BMW theoretically having 10^{32} different models in 2004, see Meyr (2004), which in 2020 is even higher with the release of new vehicle types. Even though not all different models will be made, the customers expect short lead times and personal customization on their product. Nevertheless the factory must keep the production rate and quality high while maintaining low assembly costs, see Bukchin et al. (2002). In addition the flexibility of the assembly line should be high in order to quickly react to changing customer demand and customization.

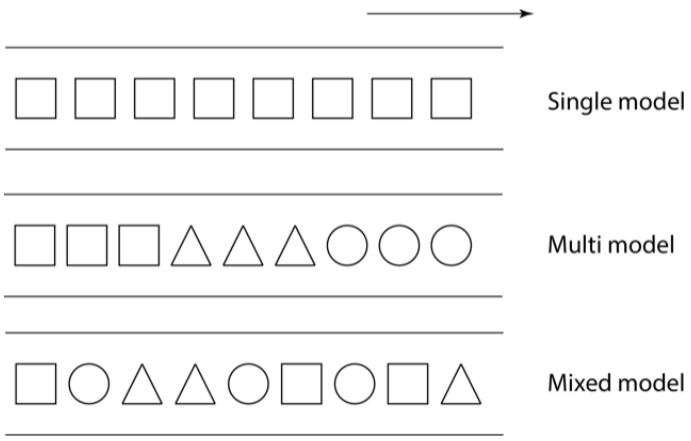


Figure 3.4: Schematic showing the 3 different types of assembly lines. The assembly lines flow from left to right and each shape represents a different model

The other two options both consider the sequencing of the different models on the assembly lines in different ways, see Boysen et al. (2007). The first option is referred to as Multi-Model Assembly Line Balancing. Here the models are sequenced in batches dependent on their demand. In this case, if the batch sizes are large enough, it may be possible to balance the line separately for each model type, which can in turn also lead to some other problems. However, as mixed-model assembly lines are not relevant for the automotive industry, further discussion regarding this subject is left out.

The third and most relevant is Mixed Model Assembly Line Balancing (MMALB). Here models are randomly sequenced on the line according to their demand. The models that are produced are typically a variant of the same base model. Nevertheless, even this restriction has been subject to change over the last years, where even completely different models are still produced on the same assembly line so that the companies become even more versatile. It quickly becomes clear the MMALBP gives rise to many problems that need to be solved to come to a feasible (and preferably optimal) line balance.

The first major challenge is the difference in task times between the different variants. Installation of a certain option may be more intensive than another option which leads to high variety in station times. On occasions this may cause the takt time to be exceeded. This is not a big problem if it occurs sporadically, though if a lot of high intensive models succeed each other a large overload occurs. This has to be counteracted by either halting the line or using an extra operator, both of which are expensive solutions and therefore highly undesired. Another way to counteract this problem is by adjusting the sequence of the models in the line, known as Car Sequencing (CS) or Mixed Model Sequencing (MMS).

Simultaneously balancing and sequencing an assembly line can be viable. However, due to the difference in planning horizons (balancing having a long term horizon and sequencing short term) more problems have to be overcome. The exact model mix is difficult to predict on a long term basis as this can already vary daily. As a result, the line balancing is done on a prediction of the demand for the main models (i.e., the various smaller options a model can have are not taken into account). As the goal is to come to a prediction of the number of operators needed to combat future demand, this assumption is reasonable. If the objective is to balance the lines on a short term bases, where the sequence of the models is known, then the lines can be balanced using a simulation based approach, see e.g. Tiacci (2015).

Strict Restriction

A second way to balance the MMAL is through strict restrictions of the takt time. In this case the takt time must be met for every model, independent of its demand. While this relaxes most of the problems as described in the previous sections, it can lead to a high amount of inefficiency especially when model demands are low but task times are high. This in turn also lead to the use of more operators along the line and in turn to higher running costs.

Example 2: Strict Restrictions *Consider 4 models labeled {A,B,C,D} each with varying demand α_m, with $\sum \alpha_m = 1$, see Figure 3.5. Model A is the only model that has no idle time as the sum of all its tasks is equal to the takt time. If the demand for model A is low, $\alpha_A = 0.1$, and the rest of the models have equal demand ($\alpha_B = \alpha_C = \alpha_D = 0.30$) then the average efficiency of the line is equal to 0.625. Considering the fact that the average efficiency is desired to be around 0.95, this method of balancing an assembly line is undesirable.*

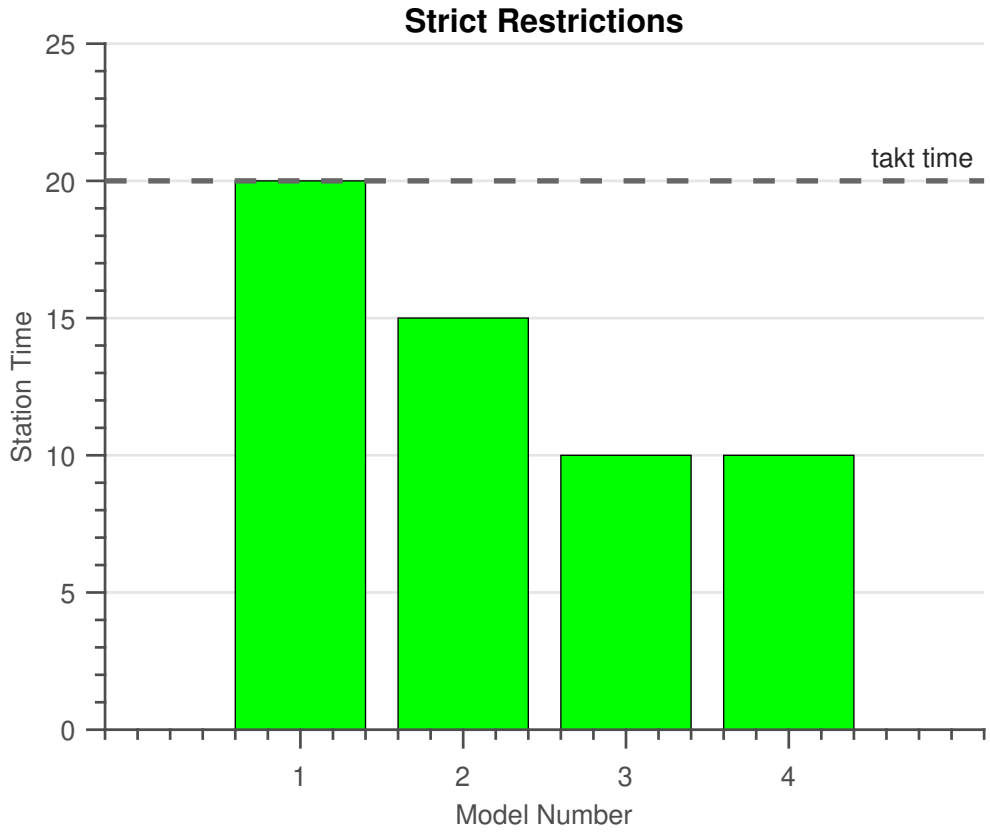


Figure 3.5: Example of strict takt time restriction considering 4 models

Joint Precedence

A common approach to balancing MMALB is that by considering the MMALB as a single model line. In this case a joint precedence graph is created that contains all the tasks of each model, see e.g. Boysen et al. (2009) and Yang et al. (2013). The task times are then averaged according to the demand of each model according to the following equations:

$$\bar{\tau_j} = \sum_{n \in N} p_n \tau_{j,n}, \tag{3.1}$$

where $\bar{\tau_j}$ is the weighted duration of task j, N is the set of models that need to be produced, p_n the demand for model n and $\tau_{j,n}$ the task duration of task j for model n. An example of a joint precedence graph can be seen in Figure 3.6. For example, task 3 is unique for model 2 and results in a processing time of $2 \cdot 0.5 + 0 \cdot 0.5 = 1$ if both models have equal demand. This however causes large overloads to occur in stations if the demand for a model is low but the task time is high (e.g., if a model's demand is 5% and its task time is 50 then its weighted task time is 2.5). This also causes a large difference in station times between the models, which in turn can block the line if two task intensive models are launched down the line consecutively. A way to counteract this problem is by allowing common tasks (i.e., tasks that are done regardless of the model) to be executed on different workstation instead of the same, see Becker and Scholl (2006). This, however, is an undesired option as, e.g., more equipment has to be placed at each workstation. Another method that is most commonly employed in the automotive industry is the use of the so called skip policy (see Scholl (1999)). Here the group coordinator takes over whenever the normal operator alongside the assembly line cannot finish their workpiece before the end of the workstation. In most cases this will not lead to extra costs and is therefore more desirable (however not always optimal).

The downside, however, to previously described method is when sequence dependent setup times are also considered. Consider three tasks with corresponding task duration, forward and backwards setup times as given in Figure 3.2. Two variants have to be produced whereby task 1 is a common task, task 2 is exclusive for variant 1 and task 3 is exclusive for variant 2. The sequence $\{1, 2, 3\}$ is still the optimal sequence in this case if the MMAL is considered as a single model line. However, this would lead to incorrect setup times being taken into account. Forward setup π_{j_2,j_3} would be added to the station time but this setup time does not exist as j_2 and j_3 are exclusive to variants v_1 and v_2 respectively. Furthermore, backward setup μ_{j_2,j_1} is not taken into account despite this setup existing for variant v_1 (similarly π_{j_1,j_3} is not taken into account for variant v_2).

Averaging Station Loads

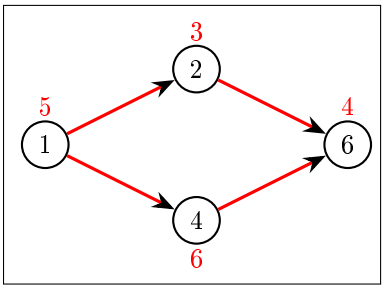
A solution to the previous described problems is to combine and adjust joint precedence and strict restrictions. The assembly is balanced for every model independently, whereby the takt time constraint is relaxed. The takt time then does not have to be met for every model, however the average station load should be less than or equal to the takt time. Then, if the station load at workstation w and operator o for model n is denoted as $t_{w,o,n}$, then the following inequality must hold:

$$\sum_{n \in N} \alpha_n \cdot t_{w,o,n} \leq \mathcal{T}. \tag{3.2}$$

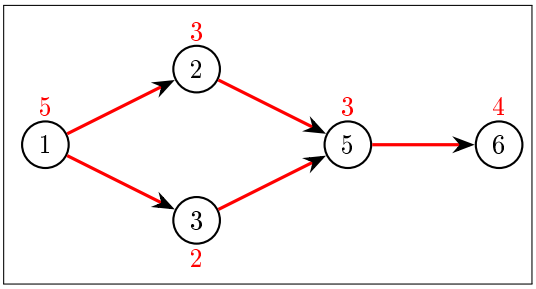
In contrast, another strict restriction is introduced. When an operator exceeds the takt time at his/her respective workstation, the operator may not be able to use certain equipment or material due to the positioning of these pieces. For that reason the operators station boundaries $\beta_{w,o}$ are introduced, denoting the distance between the start of the operators workstation and the maximum distance the operator can vary from this point. If the speed of the line is normalized to 1 (i.e., the length of a workstation is equal to the takt time), then the following inequality must hold for every operator:

$$t_{w,o,n} \leq \beta_w. \tag{3.3}$$

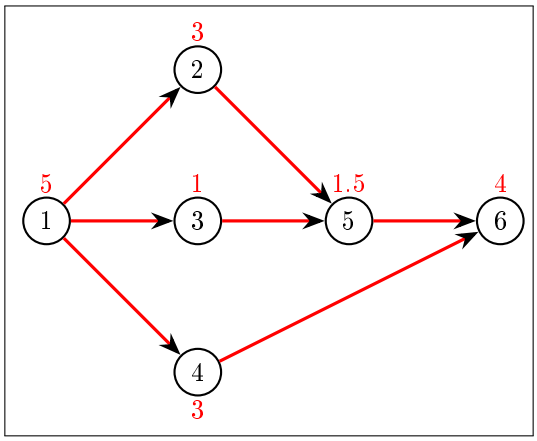
By doing this no setup times are missing in the balance and the station loads are described correctly. However a similar problem as described in section 3.1.4 is still present. A model that creates a large station load with a large demand may cause a large variance in takt time. To combat this, the method of horizontal balancing, as introduced by Merengo et al. (1999), can be used. The aim of horizontal balancing is to smooth out the work load at every mated station (w, o) over all models. Emde et al. (2010) examined different objective functions related to horizontal balancing. They concluded that comparing the station times of all models to the takt time typically leads to the best



(a) Precedence graph of Model 1



(b) Precedence graph of Model 1



(c) Precedence graph of Model 1 and 2

Figure 3.6: Example of a joint precedence diagram. All common and unique tasks between models 1 and 2 are combined to form a single precedence graph that is used to solve the ALBP. Both models have equal demand (0.5), therefore, the processing times of the tasks in the joined graph are averaged according to 3.1

results. Furthermore, if demand information can be given with a high amount of accuracy, then a weighted objective function should be used. The following two horizontal balancing approaches are therefore introduced, depending on the planning horizon:

$$\min \sum_{n \in N} \sum_{w \in W} \sum_{o \in O} |t_{w,o,n} - \mathcal{T}| \tag{3.4}$$

$$\min \sum_{n \in N} \sum_{w \in W} \sum_{o \in O} \alpha_n \cdot |t_{w,o,n} - \mathcal{T}|, \tag{3.5}$$

where (3.4) should be used for long term planning and (3.5) for short term planning where demand is certain. Minimizing the variance can result in less overall work overload for an operator (i.e., the total amount of time an operator exceeds the takt time). In addition, horizontal smoothing also helps in counteracting the volatile demand of the automotive market. Demand changes can cause line balances to become infeasible, as the average station time can start to exceed the takt time. The lower the difference between the station times and the takt time, the less the demand influences the overall average station load. It is worth noting that the higher the uncertainty in demand forecasts, the lower the effect of horizontal balancing is.

Lastly, Emde et al. (2010) concluded in the case of high product variety that separate model demand becomes completely insignificant, the concept of vertical balancing might work better. Vertical balancing aims at evenly balancing the workloads among all stations by comparing the

station time to the processing times of all models per station, in other words:

$$\sum_{w \in W} |t^*_{w,o} - \bar{t}|, \tag{3.6}$$

where $t^*_{w,o}$ is that average station time of workstation k and $\bar{t}$ the overall station time according to (3.7) and (3.8) respectively.

$$t^*_{w,o} = \sum_{n \in N} \alpha_n \cdot t_{w,o,n} \qquad \forall w \in W, o \in O, \tag{3.7}$$

$$\bar{t} = \frac{1}{|W|} \sum_{w \in W} \frac{1}{|O|} \sum_{o \in O} t^*_{w,o}. \tag{3.8}$$

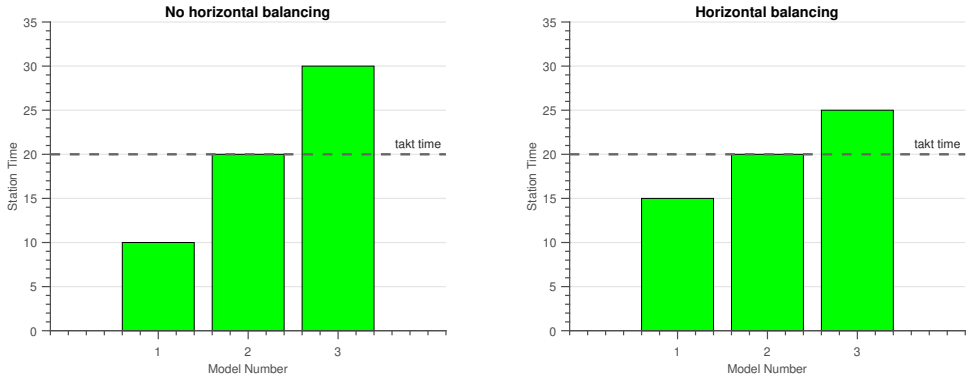
In summary, while considering mixed-model assembly lines, the following assumptions must hold:

Assumption 12 Common tasks between models must be processed on the same workstation and the same operator.

Assumption 13 The average station load across each model at every mated station (w, o) should be less than or equal to the takt time.

Assumption 14 The maximum station load observed by a single model should be less than the operators boundaries.

Assumption 15 The total workload at every mated station should be balanced out over every model (*horizontal balancing*) or balance out over all stations (*vertical balancing*) dependent on the planning horizon and demand uncertainty.



(a) No horizontal balancing objective **(b)** Horizontal balancing objective

Figure 3.7: Example showing difference between two solutions for a single mated workstation (w, o). If no horizontal balancing is applied the variance between different models is large, however, if horizontal balancing is applied this variance can be decreased. This in turn causes less overall work overload.

3.2. Assembly Line Sequencing

In this section, a solution to the Assembly Line Sequencing Problem (ASLP) is given. The current method as employed at VDL Nedcar is Car Sequencing. An overview of how the rules related to CS are generated is given, as well as the solution method for the ASLP.

3.2.1. Car Sequencing

One of the two methods for the sequencing of assembly lines in the automotive industry is CS, as has been briefly described in Section 2.3. The second method is Mixed Model Sequencing (see Golle et al. (2014)), however this is out of the scope of this project. CS is based on certain sequencing rules that depend on the results of the ALBP. The objective of CS is to sequence all N different models while following rules related to options these models can have (e.g., an a/c or sunroof). It can easily be seen that the more station times are distributed evenly across the entire assembly line, the less options require rules. In the remainder of this section two different methods are proposed to define options and rules.

Currently, only two approaches for generating sequencing rules have been formulated. Sequencing rules are typically in the form of $H_v : Q_v$, meaning that out of every Q_v sequenced models only H_v of these models can contain variant v. The first one is the BY-approach as given by Bolat and Yano (1992). The basis of these sequencing rules is to calculate a number of models Q_o that contain an option o that can be sequenced successively before a number of models not containing option o has to be sequenced to shift the operator back to its left hand side border. However, as discussed by Golle et al. (2014) this method can cause feasible sequences to be ignored. The reasoning behind this is that an operator does not necessarily have to return to his left hand border, as long as the right hand border is not exceeded. Golle et al. (2014) therefore propose a new rule, referred to as the MSR-approach (mixed Sequencing Rule). Instead of generating one rule per option, the MSR approach generates multiple per option, which can in turn lead to a more accurate sequence where the total work overload is minimized. It is assumed that exactly one variant $v \in V$ is processed at every mated workstation (w, o). The minimum and maximum number of successive models that consists of variant v that can be sequenced is then:

$$q_v^{min} = \left\lfloor \frac{\beta_w^v - T}{p_v^+ - c} \right\rfloor \tag{3.9}$$

$$q_v^{max} = \left\lfloor \frac{A(c - p_v^-) + (\beta_w^v - T)}{p_v^+ - p_v^-} \right\rfloor, \tag{3.10}$$

where β_w^v is the length of the workstation w variant v is produced on, p_v^+ is the processing time of models with variant v, p_v^- is the processing time of models not containing v and A is the length of the sequence. Furthermore it is assumed that $p_v^- < T < p_v^+ \leq l_w^v$. Finally, the sequencing rules can be generated as follows:

$$H_v^{q_v - q_v^{min}+1} = q_v \tag{3.11}$$

$$Q_v^{q_v - q_v^{min}+1} = H_v^{q_v - q_v^{min}+1} + \left\lceil \frac{q_v(p_v^+ - c) - (\beta_w^v - p_v^+)}{c - p_v^-} \right\rceil, \tag{3.12}$$

with $q_v \in [q_v^{min}; q_v^{max}]$. The last term of (3.12) calculates the amount of models not containing v that need to be processed before another model that contains v can again be processed. In addition a secondary rule for every variant is proposed. It can be useful to distribute every model over the line evenly(e.g., this can help with material planning). For every variant an extra rule is created equal to:

$$H_v = 1 \tag{3.13}$$

$$Q_v = \left\lfloor \frac{\sum_{v \in V} \alpha_v \cdot A}{\alpha_v \cdot A} \right\rfloor. \tag{3.14}$$

Lastly, a weight can be assigned to each sequencing rules. A common way to weigh each rule is to assign a weight equal to the amount of time a variant v exceeds the takt time, in other words:

$$\lambda_v = p_v^+ - T, \tag{3.15}$$

where λ_v is the work overload induced by variant v.

3.3. Summary

In this chapter a description of the model needed to solve the assembly line balancing problem was provided. First, the main problem referred to as the ALBP was described, including all the assumptions necessary to solve it. Then, additional constraints were added to the model that were previously found in Chapter 2. Different ways to incorporate these constraints were described and were added to the main model. Lastly, an additional problem known as the Car Sequencing Problem was described.

In the next chapter the assumptions that were formulated in this chapter are translated to a Mixed Integer Linear Program that can be used to solve the ALBP and the CSP.

CHAPTER 4

Mixed Integer Linear Program

In this chapter the Mixed Integer Linear Program (MILP) models are presented for the different models previously described in Chapter 3. First, a short description of a MILP is given in section 4.1. Then the necessary input data, sets and variables needed to solve both MILP models are presented. Finally, the MILP is presented respectively for the ALBP and CS.

4.1. MILP Background

In Mixed Integer Linear Programming some of the design variables that are used are constrained to be integers, while others are not. Furthermore, all variables are linearly dependent on each other. The optimization problem can be written in the following standard form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & A_{eq} x = b_{eq} \\ & Ax \leq b \\ & lb \leq x \leq ub \\ & x_i \in \mathbb{Z} \\ & x_j \in \{0, 1\} \end{aligned} \tag{4.1}$$

where x are decision variables, A_{eq} and A are $m \times n$ input matrices and c, b_{eq}, b, lb and ub are input vectors. MILP can be solved using exact methods such as the branch and bound, cutting plane or, a combination of both, referred to as the branch and cut method.

In addition to the above formulation, the 'big-M' notation is also used. This notation is used to penalize certain constraints if they are not met by adding a sufficiently large positive value M. E.g., if two variables labeled a and b both are equal to 1, then variable c must be 0:

$$\begin{aligned} M \cdot (2 - a - b) &\geq c \\ c &\geq 0. \end{aligned}$$

4.2. Input Data

The required input data needed to solve the various MILP models is given below:

- J number of tasks that need to allocated to the assembly line with task $j \in \{1, \ldots, J\}$
- W number of mated workstations with workstation $w \in \{1, \ldots, W\}$
- $\mathcal{T}$ is the Tact time
- O number of operators available at every mated station with operator $o \in \{1, \ldots, O\}$

- N number of models with model $n \in \{1, \ldots, N\}$
- V number of variants with variant $v \in \{1, \ldots, V\}$
- A number of slots in the production cycle with slot $a \in \{1, \ldots, A\}$
- Z number of zones with zone $z \in \{1, \ldots, Z\}$
- M a very large positive number
- ϵ a very small positive number

Furthermore, the following matrices and sets are required:

- β_w maximum length of workstation w
- $\mathcal{P}_j(\mathcal{P}_j^*)$ set of all (direct) predecessors of task j
- $\mathcal{S}_j(\mathcal{S}_j^*)$ set of all (direct) successors of task j
- $\mathcal{Z}_z$ set containing the tasks allocated to zone z
- $\tau_{j,n}$ is a $J \times N$ matrix denoting the task time of task j for model n
- $\pi_{j_1,j_2,n}$is a $J \times J \times N$ matrix denoting the forward setup time from task j_1 to task j_2 for model n
- μ_{j_2,j_1,n_2,n_1} is a $J \times J \times N$ matrix denoting the backward setup time from task j_2 to task j_1 from model n_2 to model n_1
- α_n is the fraction of the demand of model $n \in \{1, ..N\}$
- $\mathcal{D}$ is a set of tasks j that have to be assigned to station (w, o)
- $\mathcal{K}$ is a set of task pairs (j_1, j_2) that have to be assigned to station w on operators (s_1, s_2) with $s_1 \neq s_2$
- $\Omega_{j,n}$ is a $J \times N$ matrix stating if the completions time of a task j is larger than 0 for a model n, i.e.,:
  $$\Omega_{j,n} = \begin{cases} 1, & \text{if the completion time of task } j \text{ for model } n \text{ is larger than 0} \\ 0, & \text{otherwise} \end{cases}$$
- $\Psi_{v,n}$ is an $V \times N$ matrix denoting if a model n contains variant v, i.e.,:
  $$\Psi_{v,n} = \begin{cases} 1, & \text{if model } n \text{ contains variant } v \\ 0, & \text{otherwise} \end{cases}$$
- $\mathcal{R}$ is a set of $H_v : N_v$ sequencing rules stating that H_v out of N_v successively sequenced models can contain variant v

4.2.1. Design Variables

The following design variables are used in the MILP:

- $x_{j,w}^{JW} = \begin{cases} 1, & \text{if task } j \text{ is assinged to workstation } w \\ 0, & \text{otherwise} \end{cases}$
- $x_{j,o}^{JO} = \begin{cases} 1, & \text{if task } j \text{ is assinged to operator } o \text{ of a station} \\ 0, & \text{otherwise} \end{cases}$

- $x^{JJ}_{j_1,j_2} = \begin{cases} 1, & \text{if task } j_1 \text{ is assigned to an earlier position than task } j_2 \text{ and } j_1 \neq j_2 \text{ in a mated station} \\ 0, & \text{otherwise} \end{cases}$
- $x^{AN}_{a,n} = \begin{cases} 1, & \text{if model } v \text{ is assigned to postion } a \text{ of the production cycle} \\ 0, & \text{otherwise} \end{cases}$
- $y^{JJN}_{j_1,j_2,n} = \begin{cases} 1, & \text{if task } j_1 \text{ directly precedes task } j_2 \text{ for model } n \text{ in a mated station and } j_1 \neq j_2 \\ 0, & \text{otherwise} \end{cases}$
- $f^{JN}_{j,n} = \begin{cases} 1, & \text{if task } j \text{ is assigned to the first position for model } nv \\ & \text{in a mated station and } \Omega_{j_1,n} > 0 \\ 0, & \text{otherwise} \end{cases}$
- $l^{JN}_{j,n} = \begin{cases} 1, & \text{if task } j \text{ is assigned to the last position for model } n \\ & \text{in a mated station and } \Omega_{j_1,n} > 0 \\ 0, & \text{otherwise} \end{cases}$
- $u^{WO}_{w,o} = \begin{cases} 1, & \text{if mated station } (w,o) \text{ is utilized} \\ 0, & \text{otherwise} \end{cases}$
- $u_w = \begin{cases} 1, & \text{if workstation } w \text{ is utilized} \\ 0, & \text{otherwise} \end{cases}$
- $c^{JN}_{j,n}$ completion time of task j for model v in its mated station
- $t^{WON}_{w,o,n}$ station time of workstation w for operator o for model v
- $s^{WOA}_{w,o,a}$ starting time of operator o on workstation w for production cycle a
- $q^{WOA}_{w,o,a}$ work overload of operator o on workstation w for production cycle a
- b^{V}_{v} the amount of violation made for variant v

4.3. Assembly Line Balancing Problem

The MILP for the ALBP is formed as station and scheduling problem, which is also described in the papers of Esmaeilbeigi et al. (2016) and Yang and Cheng (2019). Station based formulations use binary decision variables to assign tasks to workstations (i.e., station information is directly available). Schedule based formulations output the sequence of the tasks, disregarding information like which task is assigned to which station. Due to the nature of the problem a combination of both methods is proposed for ease of definition. The total number of variables for the ALBP is bound by $\mathcal{O}(J^2)$.

4.3.1. Constraints

The constraint for the MILP can be divided into 6 separate categories; Task assignment, precedence, sequence, setup, capacity and station assignment constraints.

Task Assignment Constraints

Each task should only be assigned to a single workstation and a single operator:

$$\sum_{w \in W} x_{j,w}^{JW} = 1 \qquad \forall j \in J, \tag{4.2}$$

$$\sum_{o \in O} x_{j,o}^{JO} = 1 \qquad \forall j \in J. \tag{4.3}$$

If a task j has to be assigned to a mated station (w, o) then $x_{j,w}^{JW}$ and $x_{j,o}^{JO}$ should equal 1:

$$x_{j,w}^{JW} - x_{j,o}^{JO} = 0 \qquad \forall (j, (w, o)) \in \mathcal{D}. \tag{4.4}$$

If 2 tasks have to be carried out simultaneously at opposite operators of the same station j then the tasks should be assigned to the same station and on different operators:

$$x_{j_1,w}^{JW} - x_{j_2,w}^{JW} = 0 \qquad \begin{array}{l} \forall (j_1, j_2) \in \mathcal{K}, \\ \forall w \in W, \end{array} \tag{4.5}$$

$$x_{j_1,o_1}^{JO} - x_{j_2,o_2}^{JO} = 0 \qquad \begin{array}{l} \forall (j_1, j_2) \in \mathcal{K}, \\ \forall (o_1, o_2) \in O, \\ o_1 \neq o_2. \end{array} \tag{4.6}$$

To operators on the same workstation cannot be assigned tasks that occupy the same zone:

$$M \cdot (2 - x_{j_1,w}^{JW} - x_{j_2,w}^{JW}) \geq x_{j_1,o_1}^{JO} - x_{j_2,o_2}^{JO} \qquad \begin{array}{l} \forall z \in Z, \\ \forall (j_1, j_2) \in \mathcal{Z}_z, \\ \forall w \in W, \\ \forall o \in O. \end{array} \tag{4.7}$$

Precedence Constraints

All tasks must follow the precedence constraints, i.e. a task cannot be assigned to a workstation, unless all of its predecessors have been assigned to an earlier or the same workstation. Note that for the precedence constraints it is not necessary that a task's predecessors have also been assigned to the same operator of a workstation.

$$\sum_{y_1 \in W} y_1 \cdot x_{j_1,y_1}^{JW} \leq \sum_{y_2 \in W} y_2 \cdot x_{j_2,y_2}^{JW} \qquad \begin{array}{l} \forall j_1 \in J, \\ j_2 \in \mathcal{P}_{j_1}^{*}. \end{array} \tag{4.8}$$

Sequence Constraints

2 tasks that are assigned to the same workstation are also assigned in a specific sequence (i.e., the order in which the tasks are executed in the workstation). This is denoted by the variable x_{j_1,j_2}^{JJ}. If tasks j_1 and j_2 are assigned to the same mated station (w, o) and they are not direct

predecessors of one another, then one of those tasks must be appointed to an earlier position in the mated station (i.e, only variable $x^{JJ}_{j_1,j_2}$ or $x^{JJ}_{j_2,j_1}$ can be equal to 1), this is ensured by the following 2 constraints:

$$x^{JJ}_{j_1,j_2} + x^{JJ}_{j_2,j_1} + M \cdot (4 - x^{JW}_{j_1,w} - x^{JW}_{j_2,w} - x^{JO}_{j_1,o} - x^{JO}_{j_2,o}) \geq 1 \qquad \begin{array}{r} \forall j_1 \in J, \\ \forall j_2 \in J - (\mathcal{P}^*_{j_1} \cup \mathcal{S}^*_{j_1}), \\ \forall w \in W, \\ \forall o \in O, \end{array} \tag{4.9}$$

$$x^{JJ}_{j_1,j_2} + x^{JJ}_{j_2,j_1} - M \cdot (4 - x^{JW}_{j_1,w} - x^{JW}_{j_2,w} - x^{JO}_{j_1,o} - x^{JO}_{j_2,o}) \leq 1 \qquad \begin{array}{r} \forall j_1 \in J, \\ \forall j_2 \in J - (\mathcal{P}^*_{j_1} \cup \mathcal{S}^*_{j_1}), \\ \forall w \in W, \\ \forall o \in O. \end{array} \tag{4.10}$$

If task j_1 is an immediate predecessor of task j_2 and they are allocated to the same mated station (w, o), then j_1 must also precede task j_2 in that mated station:

$$x^{JJ}_{j_1,j_2} + M \cdot (4 - x^{JW}_{j_1,w} - x^{JW}_{j_2,w} - x^{JO}_{j_1,o} - x^{JO}_{j_2,o}) \geq 1 \qquad \begin{array}{r} \forall j_2 \in J, \\ \forall j_1 \in \mathcal{P}_{j_2}, \\ \forall w \in W, \\ \forall o \in O. \end{array} \tag{4.11}$$

If a task j_2 precedes task j_3 in a station and consequently task j_1 precedes task j_2 in the same station task, then j_3 must therefore also precede j_1 in the same mated station. This is also referred to as a transitivity constraint.

$$x^{JJ}_{j_1,j_3} + M \cdot (2 - x^{JJ}_{j_1,j_2} - x^{JJ}_{j_2,j_3}) \geq 1 \qquad \forall (j_1, j_2, j_3) \in J. \tag{4.12}$$

If 2 tasks are not assigned to the same workstation or operator then there cannot be any sequence dependent relations between them:

$$\begin{array}{l} M(1 - x^{JJ}_{j_1,j_2}) \geq x^{JW}_{j_1,w} + x^{JW}_{j_2,w}, \\ M(1 - x^{JJ}_{j_1,j_2}) \geq x^{JW}_{j_2,w} + x^{JW}_{j_1,w} \end{array} \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall w \in W. \end{array} \tag{4.13}$$

$$\begin{array}{l} M(1 - x^{JJ}_{j_1,j_2}) \geq x^{JO}_{j_1,o} + x^{JO}_{j_2,o}, \\ M(1 - x^{JJ}_{j_1,j_2}) \geq x^{JO}_{j_2,o} + x^{JO}_{j_1,o} \end{array} \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall o \in O. \end{array} \tag{4.14}$$

Setup Constraints

If for any model v its task time for a task j_1 is equal to 0, then no forward setup can occur for the task of that model:

$$y^{JJN}_{j_1,j_2,n} - M \cdot (\Omega_{j_1,n_1} \cdot \Omega_{j_2,n_2}) \leq 0 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall (n_1, n_2) \in N. \end{array} \tag{4.15}$$

A task j_1 cannot be an immediate follower of another task j_2 in both forward and backward direction (i.e., if task j_1 is the first task in a station and task j_2 is the last task in a station, then there cannot be a forward setup from task j_2 to task j_1).

$$y^{JJN}_{j_2,j_1,n} - M \cdot (2 - f^{JN}_{j_1,n} - l^{JN}_{j_2,n}) \leq 0 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall n \in N. \end{array} \tag{4.16}$$

Each task j in a mated station (w, o) can have at most one direct follower in forward direction:

$$\sum_{j_2 \in J} y^{JJN}_{j_1,j_2,n} \leq 1 \qquad \begin{array}{l} \forall j_1 \in J, \\ \forall n \in N. \end{array} \tag{4.17}$$

Only one forward setup can occur between any task pair in a mated station (w, o):

$$y^{JJN}_{j_1,j_2,n} + y^{JJN}_{j_2,j_1,n} \leq 1 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall n \in N. \end{array} \tag{4.18}$$

A forward setup can only occur between tasks that have been assigned to the same mated station (w, o).

$$x^{JJ}_{j_1,j_2} \geq y^{JJN}_{j_1,j_2,n} \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall n \in N. \end{array} \tag{4.19}$$

If task j_1 directly precedes task j_1 in a mated station (w, o) for a model v, then a forward setup should exist:

$$\begin{array}{l} y^{JJN}_{j_1,j_2,n} + M \cdot (6 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o} - x^{JW}_{j_2,w} - x^{JO}_{j_2,o} - \Omega_{j_1,n} - \Omega_{j_2,n}) + \\ M \cdot |\sum_{i \in J} x^{JJ}_{j_1,i} \cdot \Omega_{i,n} - \sum_{k \in J} x^{JJ}_{j_2,k} \cdot \Omega_{k,n} - 1| \geq 1 \end{array} \qquad \begin{array}{l} \forall j_1 \in J, \\ \forall j_2 \in J - \mathcal{P}_{j_1}, \\ \forall w \in W, \\ \forall o \in O, \\ \forall n \in N. \end{array} \tag{4.20}$$

The first and last task in a station load can be determined as follows; the first task in a station is a task that no other tasks have any forward setups to and similarly the last task in a station is

the task that does not have any forward setups to another task.

$$f^{JN}_{j_1,n} + M \cdot (3 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o} - \Omega_{j_1,n}) + \sum_{j_2 \in J} x^{JJ}_{j_2,j_1} \cdot \Omega_{j_2,n} \geq 1 \qquad \begin{array}{l} \forall j \in J, \\ \forall w \in W, \\ \forall o \in O, \\ \forall n \in N, \end{array} \tag{4.21}$$

$$l^{JN}_{j_1,n} + M \cdot (3 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o} - \Omega_{j_1,n}) + \sum_{j_2 \in J} x^{JJ}_{j_1,j_2} \cdot \Omega_{j_2,n} \geq 1 \qquad \begin{array}{l} \forall j \in J, \\ \forall w \in W, \\ \forall o \in O, \\ \forall n \in N. \end{array} \tag{4.22}$$

In addition, if the previous is not true, than a task can not be first or last in a station respectively:

$$f^{JN}_{j_1,n} + y^{JN}_{j_2,j_1,n} - M \cdot (6 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o} - x^{JW}_{j_2,w} - x^{JO}_{j_2,o} - \Omega_{j_1,n} - \Omega_{j_2,n}) \leq 1 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall w \in W, \\ \forall o \in O, \\ \forall n \in N, \end{array} \tag{4.23}$$

$$l^{JN}_{j_1,n} + y^{JN}_{j_1,j_2,n} - M \cdot (6 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o} - x^{JW}_{j_2,w} - x^{JO}_{j_2,o} - \Omega_{j_1,n} - \Omega_{j_2,n}) \leq 1 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall w \in W, \\ \forall o \in O, \\ \forall n \in N. \end{array} \tag{4.24}$$

Capacity Constraints

The completion time of every task j should be larger or equal to its task time:

$$c_{j,n} \geq \tau_{j,n} \qquad \begin{array}{l} \forall j \in J, \\ \forall n \in N. \end{array} \tag{4.25}$$

The average completion time for all tasks on each workstation for every model should be less than or equal to the takt time:

$$\sum_{n \in N} \alpha_v \cdot t^{WON}_{w,o,n} \leq T \qquad \forall j \in J. \tag{4.26}$$

The station time for each model cannot exceed the line length:

$$t^{WON}_{w,o,n} \leq \beta_w \qquad \begin{array}{l} \forall w \in W, \\ \forall o \in O, \\ \forall n \in N. \end{array} \tag{4.27}$$

The completion time of every task on the last workstation should be smaller than or equal to the takt time:

$$c_{j,n}^{JN} - M \cdot (1 - x_{j,W}^{JW}) \leq T \qquad \begin{array}{l} \forall j \in J, \\ \forall n \in N. \end{array} \tag{4.28}$$

The completion time of every task including the forward setup time to the next task in the sequence can be calculated as follows:

$$c_{j_2,n}^{JN} - c_{j_1,n}^{JN} + M \cdot (1 - y_{j_1,j_2,n}^{JJN}) \geq \tau_{j_2,n} + \pi_{j_1,j_2,n} \qquad \begin{array}{l} \forall j_2 \in J, \\ \forall j_1 \in J - \mathcal{P}_{j_2}, \\ \forall n \in N. \end{array} \tag{4.29}$$

The backward setup time from the first task in the sequence to the last task should be added to the station time. However, when it is assumed that the next model in the production sequence is unknown, then it does not become clear which backward setup time should be added to the station load. Therefore the average backward setup time between all models should be added, in other words:

$$c_{j_2,n}^{JN} + \mu_{j_2,j_1} \leq t_{w,o,n}^{WON} + M \cdot (6 - f_{j_1,n_1}^{JN} - l_{j_2,n_2}^{JN} - x_{j_1,w}^{JW} - x_{j_2,w}^{JW} - x_{j_1,o}^{JO} - x_{j_2,o}^{JO}) \qquad \begin{array}{l} \forall j_1 \in J, \\ \forall j_2 \in J - \mathcal{S}_{j_1}, \\ \forall (n_1, n_2) \in N, \end{array} \tag{4.30}$$

where:

$$\mu_{j_2,j_1} = \frac{1}{N^2} \sum_{n_1 \in N} \sum_{n_2 \in N} \mu_{j_2,j_1,n_1,n_2} \qquad \forall (j_1, j_2) \in J. \tag{4.31}$$

If task j_1 precedes task j_2 in a mated station (w, o) then the finish time of task j_1 must be lower than that of task j_2.

$$c_{j_2,n}^{JN} - c_{j_1,n}^{JN} + M \cdot (1 - x_{j_1,j_2}^{JJ}) \geq 0 \qquad \begin{array}{l} \forall (j_1, j_2) \in J, \\ \forall n \in N. \end{array} \tag{4.32}$$

The completion time of a task j must be smaller than that of all its followers:

$$c_{j_2,n_2}^{JN} - c_{j_1,n_1}^{JN} + M \cdot (2 - x_{j_2,w}^{JW} - x_{j_1,w}^{JW}) \geq \tau_{j_2,n_2} \qquad \begin{array}{l} \forall j_2 \in J, \\ \forall j_1 \in \mathcal{P}_{j_2}^*, \\ \forall w \in W, \\ \forall (n_1, n_2) \in N. \end{array} \tag{4.33}$$

If task j_1 is the first task in mated station (w, o) and task j_2 is the last task in mated station $(w-1, s)$ and task j_2 is a predecessor of task j_1, then task j_1 can only start after j_2 has finished if it has exceeded the takt time.

$$c_{j_1,n_1}^{JN} + M \cdot (4 - x_{j_1,w}^{JW} - x_{j_2,w-1}^{JW} - f_{j_1,n_1}^{JN} - l_{j_2,n_2}^{JN}) \geq c_{j_2,n_2}^{JN} - T + \tau_{j_1,n_2} \qquad \begin{array}{l} \forall j_2 \in J, \\ \forall j_1 \in \mathcal{P}_{j_2}^*, \\ \forall (n_1, n_2) \in N, \\ \forall w \in W. \end{array} \tag{4.34}$$

The completion times of tasks that have to be executed simultaneously should be the same (it is assumed that the task times of both tasks are equal)

$$c^{JN}_{j_1,n} - \tau_{j_1,n} - c^{JN}_{j_2,n} + \tau_{j_2,n} = 0 \qquad \forall (j_1, j_2) \in \mathcal{K}, \; \forall n \in N. \tag{4.35}$$

Lastly, the completion time of any task and the station time cannot be negative, therefore:

$$c^{JN}_{j,n} \geq 0 \qquad \forall j \in J \; \forall n \in N, \tag{4.36}$$

$$t^{WON}_{w,o,n} \geq 0 \qquad \forall w \in w \; \forall o \in O, \; \forall n \in N. \tag{4.37}$$

Station Assignment Constraint

If any task j is assigned to a mated station (w, o) then that mated station should be active (i.e., $u^{WO}_{w,o} = 1$:

$$u^{WO}_{w,o} + M \cdot (2 - x^{JW}_{j_1,w} - x^{JO}_{j_1,o}) \geq 1 \qquad \forall j \in J, \; \forall w \in W, \; \forall o \in O. \tag{4.38}$$

If a task is assigned to a workstation operator, then that workstation must be active:

$$u^{W}_{w} + M \cdot (1 - u^{WO}_{w,o} \geq 1 \qquad \forall w \in W, \; \forall o \in O. \tag{4.39}$$

Furthermore, if no task is allocated to an operator in a workstation, then that workstation is not active:

$$u^{W}_{w} + M \cdot \sum_{o \in O} u^{WO}_{w,o} \geq 1 \qquad \forall w \in W. \tag{4.40}$$

Lastly, all workstation must be active in order:

$$u^{W}_{w+1} \leq u^{W}_{w} \qquad \forall w \in W. \tag{4.41}$$

4.3.2. Objective

The objective of the model is to minimize the number of operators and workstation used on the assembly line and to keep the station time and horizontal balancing as low as possible. Therefore the following objective function is chosen:

$$\min M \cdot \sum_{w \in W} \sum_{o \in O} u^{WO}_{w,o} + \frac{M}{10} \cdot \sum_{w \in W} u^{W}_{w} + \frac{\epsilon}{10} \cdot \sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} |t_{w,o,n} - \mathcal{T}| + \epsilon \cdot \sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} t_{w,o,n} \tag{4.42}$$

The separate terms are weighted according to their importance in the overall objective. The number of operators should dominate the other coefficients, as this is the main objective of the model. The term related to the number of workstations can be given more or less importance depending on the to be balanced line as this depends on the relation between the number of tasks, their task time and the available amount of workstations. Assembly lines with a high amount of tasks, that have high processing times and are very limited in the amount of available workstations, will have to put more emphasis on lowering the number of workstations, as the correct amount may be difficult to achieve otherwise.In this case it is chosen as the second most important factor. Decreasing station times and horizontal balancing are given the least importance. However the station times are given a slightly higher factor, as the horizontal balancing term may otherwise add idle times to the solution, which is undesired. Here, M is set to 100 and $\epsilon = \frac{1}{\mathcal{T} \cdot N}$

4.3.3. Results

The MILP for the ALBP was implemented into a Python API using IBM ILOG CPLEX as a solver. Five small test case were used from `https://assembly-line-balancing.de/`. The size and results of the test cases can be seen in Table 4.1. It becomes apparent that even for a very small case consisting of 11 tasks the computation time is very high. For the test case consisting of 31 tasks the model resulted in an *out-of-memory* error. Therefore, it can be concluded that the MILP cannot be used for actual test cases, as the computation time increases exponentially with an increasing number of tasks and models.

Table 4.1: Results of the MILP model for different test cases. The demand for each model was equal according to the number of models present in the test case. The time limit for each run was set to *7200* seconds.

Name	**#tasks**	**#models**	**takt time**	**#operators**	**#workstations**	**objective**	**gap(%)**	**CPU time (s)**
mertens	11	2	18	2	1	211.775	0	1181
mertens	11	2	18	4	3	433.325	35	7200
jackson	13	2	18	3	3	335.147	33	7200
roszieg	31	3	25	-	-	-	-	>7200

4.4. Car Sequencing

In this section the MILP for CS is given, following the explanation as previously given in section 3.2.1. First, each slot in the production cycle can only contain a single model:

$$\sum_{n \in N} x_{a,n}^{AN} = 1 \qquad \forall a \in A. \tag{4.43}$$

Secondly, the total demand must be satisfied for the entire production cycle:

$$\sum_{t \in T} x_{a,n}^{AN} = \alpha_n \cdot A \qquad \forall n \in N. \tag{4.44}$$

Lastly, the amount of violations induced by variant v that occur in a specific window need to be counted. The basis of this violation rule is that each excessive variant in a window leads to an extra violation. The length of the window, as described in section 3.2.1, runs from position a in the total sequence, to position $a + G_v + 1$ in the sequence. The amount of models in this window that contains variant v is summed, and the difference with the amount of models allowed (H_v) according to the CS-rule, is calculated. However, there can be less models that contain variant v in the window, resulting in a negative amount of violations. Therefore the maximum between

the calculated amount of violations and zero has to be taken. Finally, the total sum over the production sequence, including the extra added production slots, is used to count the total amount of violations variant v incurs.

$$b_v^V = \sum_{a=H_v-G_v+2}^{A-H_v} \max\left(\sum_{a'=a}^{a+G_v+1} \sum_{n \in N} \Psi_{v,n} \cdot x_{a',n}^{AN} - H_v, 0 \right) \qquad \forall v \in V. \tag{4.45}$$

In order to linearize (4.45), an extra variable $\rho_{a,v}$ is introduced, that calculates the number of violations that are incurred:

$$\rho_{a,v}^{AV} = \left(\sum_{a'=a}^{a+G_v+1} \sum_{n \in N} \Psi_{v,n} \cdot x_{a',n}^{AN} - H_v, 0 \right) \qquad \forall v \in V, \quad \forall a \in A. \tag{4.46}$$

Then, the number of violations that are incurred by variant v is:

$$b_v^V = \sum_{a=H_v-G_v+2}^{A-H_v} \rho_{a,v}^{AV} \qquad \forall v \in V. \tag{4.47}$$

Finally, the number of violations cannot be negative:

$$b_v^V \geq 0 \qquad \forall v \in V, \tag{4.48}$$

$$\rho_{a,v}^{AV} \geq 0 \qquad \forall a \in A, \quad \forall v \in V. \tag{4.49}$$

Note that (4.49) is needed to ensure that there cannot be a negative amount of violations. The objective of the model is to minimize the amount of violations made. The total model therefore becomes:

$$\begin{aligned} &\min && \sum_{v \in V} \lambda_v \cdot b_v^V \\ &\text{s.t.} && (4.43), (4.44), (4.46), (4.47), (4.48), (4.49) \end{aligned}$$

4.5. Summary

In this chapter a MILP model to solve the ALBP and CSP was given. The solution methods and assumptions as proposed in Chapter 3 were translated to linear constraints. Also, some extra constraints were given to account for the additional constraints that may be present in a real world assembly line, such as tasks that require multiple operators at the same time. The MILP models were also tested on test sets. The ALBP proved to need a large amount of time to be solved for a small test case and would therefore not be of practical used for a real world case. For this reason, in the next chapter and alternative to the MILP is presented that makes it possible to solve the problem in a more time efficient manner.

CHAPTER 5

Algorithm

As previously described in Chapter 4, the ALBP is too large to be solved optimally in reasonable time using exact methods such as branch-and-bound. Therefore, in this chapter, an algorithm is proposed to approximate the optimal solution within a more reasonable time frame. First, an explanation of a Genetic Algorithm (GA) is provided. The GA forms the basis of the proposed solution method. Next, a meta-heuristic is provided to generate initial sequences for the GA. Lastly, an algorithm is given to divide the sequence of generated tasks among the available workstation, and operators.

5.1. Genetic Algorithm

The basis of a Genetic Algorithm (GA) is the evolution of a solution into a better solution. The genetic algorithm is an algorithm that follows the laws of reproduction and natural selection. During reproduction **genes** (traits) of the parents are combined to form a new set of genes for the offspring. The combination of the genes into a string is referred to as a **chromosome**. A chromosome therefore contains all the traits of a living organism. During reproduction the genes of the parents are crossed, referred to as **crossover**. After the parents chromosomes are crossed there is a probability that the offspring undergoes a **mutation** (caused by errors from copying the parents genes). Finally the **fitness** of the organism is how successful they are in life.

This algorithm has been used to solve various ALBPs over the past decades (see e.g., Tiacci (2015), Sabuncuoglu et al. (2000), Chen et al. (2019) and Kim et al. (2009)). Even though it is not a state-of-the-art meta-heuristic, it has been proven to be very effective in solving the ALBP and generating promising results. Typically, past literature revolves around the Type-II problems, i.e. minimizing the takt time. However all these problems can easily be adapted to solve the Type-I problem, i.e. minimizing the number of workstations used (by adapting the fitness function).

Due to the sequence dependencies caused by the setup times, it becomes favorable to allow a high variety of sequences to remain in the population, allowing for possibly more effective crossovers to occur between 2 parent chromosomes. Therefore a high number of individuals should be chosen to be carried over to the next generation.

The basis of the GA can be summarized in the following steps:

1. Generate an initial population.
2. Assign a fitness score to each individual.
3. Use tournament method to select individuals to perform crossovers.
4. Select individuals to perform mutation on.

5. Collect newly generated individuals and original individuals into a new population and assign fitness scores. Select $\mathcal{N}$ individuals to be carried over to the next generation.

6. Terminate if a stopping criterion is met (time limit, number of generations ($\#gen$), fitness score), otherwise repeat from step 3.

The GA also depends on multiple input parameters, a summary of these can be found in table 5.1.

Table 5.1: Parameters of the GA

Parameter	Description
p_m	Probability of mutation
p_c	Probability of crossover
$\#gen$	Number of generations
$\mathcal{N}$	Population size
η_t	Threshold efficiency
p_t	Tournament size

5.1.1. Encoding

The first step to the GA is encoding the chromosomes (i.e., a possible solution for the problem). As described in the previous section, chromosomes are built up out of genes. For the ALBP each gene in the chromosome represents a task, e.g. task 7 would be given a gene labeled with 7. The total length of a chromosome equals the number of available tasks. The sequence of the genes in the chromosome represents the sequence (i.e., the order) of the tasks on the assembly line. See Figure 5.1 for a visual example of a chromosome. It must be noted that the order in which the tasks appear in the chromosome, do not exactly relate to the order in which the tasks are processed. Due to the fact that multiple operators can be present at every workstation, successive tasks that are present in the chromosome may start simultaneously, but at different operators. The de-coding heuristic used to define at which mated station task j is present is given in section 5.1.7.

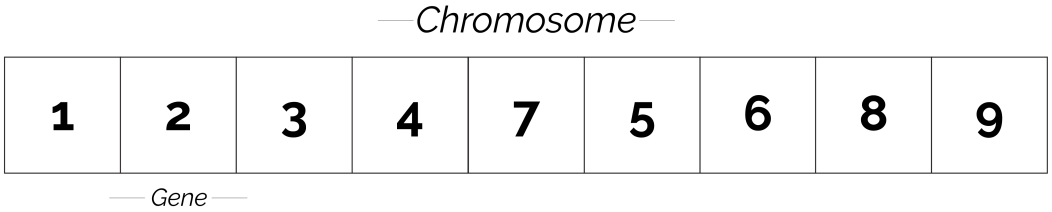


Figure 5.1: Example of a chromosome as used in the Genetic Algorithm. Each gene in the chromosome represents a task. The order in which the tasks appear in the chromosome is the order in which they are distributed over the available operators and workstations.

5.1.2. Initial Population

Due to the fact that the solutions for the ALBP are highly sequence dependent, mainly caused by the sequence dependent setup times, a diverse population can be useful for the GA. The initial population should therefore also contain a variety of unique task sequences, thus allowing for the crossover and mutation to be effective. Therefore, it has been chosen to use the method developed by Scholl et al. (2013) to generate the initial chromosomes. This method is referred to as rule-GRASP (Greedy Randomized Adaptive Search Procedure) and is a combination of previous

developed meta-heuristics from Andrés et al. (2006) and Martino and Pastor (2010). Scholl et al. (2013) propose a GRASP algorithm but, rather than choosing tasks at random which adheres to a certain threshold (a threshold is typically a calculation of the impact a task has to a certain sequence), a priority rule is chosen at random and the task with the highest priority is assigned to the sequence, similar to the method from Martino and Pastor (2010). With this method other criteria can be used to sequence the tasks among the workstations. Scholl et al. (2013) propose the following priority rules:

MT Select a task that has the highest processing time and the smallest increase of setup time:
$MT_i(k,p) = \tau_i + (\tau_{\sigma^k_{p-1,i}} + \mu_{i,\pi^k_1} - \mu_{\sigma^k_{p-1},\sigma^k_1}) \qquad \forall i \in CL$

MTS Select a task with highest processing time and the smallest increase of setup time divided by the number of available workstations:
$MTS_i(k,p) = \frac{MT_i(k,p)}{L_i(\bar{m}) - E_i + 1} \qquad \forall i \in CL$

MF Select a task that has the maximum number of direct followers:
$MF_i = |F_i| \qquad \forall i \in CL$

MS Select a task that has the minimal amount of slack (maximum value):
$MS_i = k - L_i(\bar{m}) \qquad \forall i \in CL$

MR Select any task at random.

The rule-GRASP algorithm assigns available tasks from a generated candidate list (CL) to the current constructed sequence (labeled as σ). The tasks (labeled j) in the CL are ones that can be assigned to the next position in the sequence, due to their precedence relations (i.e., tasks that do not have any predecessors or whose predecessors have already been assigned). Then a priority rule (PR) is chosen at random from a uniform distribution and each task in the CL is assigned a value corresponding to the PR. The task with the lowest PR value is then assigned to the next position in the sequence. Lastly, the CL is updated by adding the direct successors of task j (i.e., from the set $\mathcal{S}^*_j$) to the CL if all other predecessors of these tasks have also been assigned. This process is repeated until all tasks have been assigned. See algorithm 1 for a summary of the algorithm.

Algorithm 1: rule-GRASP algorithm

```
while Cl ≠ ∅ do
    select priority rule from random uniform distribution;
    for j ∈ CL do
        calculate values for chosen PR;
        add j to σ;
    end
    construct CL from S*_j;
end
```

5.1.3. Fitness Evaluation

An important factor to consider in a GA is the assignment of fitness values to the chromosomes in the (newly) generated population. The evaluation of a chromosomes fitness is based off the overall objective of the model. In the case for the ALBP, the main objective is to increase the OEE (Over Equipment Efficiency). This is achieved by reducing the number of operators working at the assembly line, in other words:

$$\rho_1 = \sum_{w \in W} O_w,$$

where O_w is the amount of operators in workstation w. In addition, two side objectives are introduced. Due to the fact that the maximum number of workstations that can be used is given, the fitness function should also include this. By doing this, solutions that generate too many workstations can be given a very high fitness value and are therefore not carried over to the next generation. It must also be noted that using less workstations than are physically available can also be desired. This could in turn lead to a better distribution of tasks throughout the entire assembly line and distribute the overall workload. In summary, the secondary fitness function becomes:

$$\rho_2 = W_{total},$$

where W_{total} is the total amount of workstation in the assembly line. Next, as stated in Section 3.1.4, to combat fluctuations in demand, horizontal balancing should also be taken into account. Therefore, the station time for each model should be known and the difference between all station times and the takt time is summed and minimized. In other words:

$$\rho_3 = \sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} |t_{w,o,n} - \mathcal{T}|.$$

Lastly, the total non-useful time (i.e., setup time and idle time) should be minimized. This can in turn help with emptying certain operators that have high amounts of non-useful time, possibly being able to reallocate certain tasks to decrease the number of operator or workstations. In addition, this will help with the horizontal balancing, as idle times are kept as low as possible and not added to the station times. This can be done by minimizing all station times. The fitness function becomes:

$$\rho_4 = \sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} t_{w,o,n}.$$

All 4 fitness are then combined into a single fitness function:

$$\rho = 100 \cdot \rho_1 + 10 \cdot \rho_2 + \frac{1}{\mathcal{T} \cdot N} \left(0.1 \cdot \rho_3 + \rho_4\right).$$

5.1.4. Crossover Children

Crossover children are created by selecting 2 parent chromosomes. The parents chromosomes can be chosen using a variety of methods such as: at random, roulette wheel selection or tournament selection. In this case the tournament selection procedure is chosen. Here, a p_t part of the population of random chromosomes is chosen (with $n > 2$) from the current population and the two with the best fitness values are chosen to form a pair of parents. This process is repeated $\#gen/2$ times.

In order to retain feasible sequences of tasks (i.e., according to the precedence relations), a 2 point crossover method is chosen. To start, two random integers ($C1$ and $C2$) between $(2, n-1)$ are chosen (with $C1 < C2$). $C1$ and $C2$ denote positions in the chromosome. Then, the head and tail portions of the first parent chromosome (P1) are copied to the first offspring (O1). The head is the genes from positions 1 up to $C1$ and the tail from positions $C2$ up to n. Then the middle section (genes between $C1$ and $C2$) of P1 is ordered in the sequence it occurs in the second parent chromosome (P2). The elements are the copies to O1. The same procedure is used to create O2, where the roles of P1 and P2 are reversed. See Figure 5.2 for an example of the 2 point crossover method.

In addition, not all parents produce an offspring with the crossover method. There is a probability, p_c with $0.5 \leq p_c \leq 1$, that two parent chromosomes produce an offspring. The value for p_c is determined through a trial-and-error approach, as different assembly lines can require different values of p_c to produce good results.

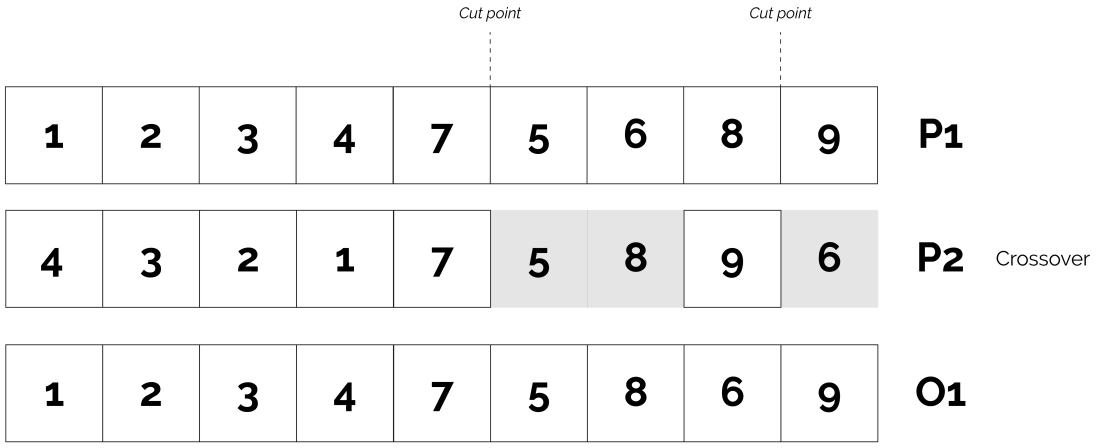


Figure 5.2: Example of the crossover operation

5.1.5. Mutation Children

The second method used to diversify the population is through mutation. Mutation children are created by first randomly selecting a gene in a parent chromosome. For this gene the latest assigned predecessor ($m1$) and earliest assigned successor ($m2$) are recorded. A random integer from $(m1, m2)$ is then chosen and the gene is inserted into this point. The probability that a mutation happens, p_m, with $0 \leq p_m \leq 0.2$, is also chosen through a trial-and-error approach. See Figure 5.3 for a visual representation of the mutation operator.

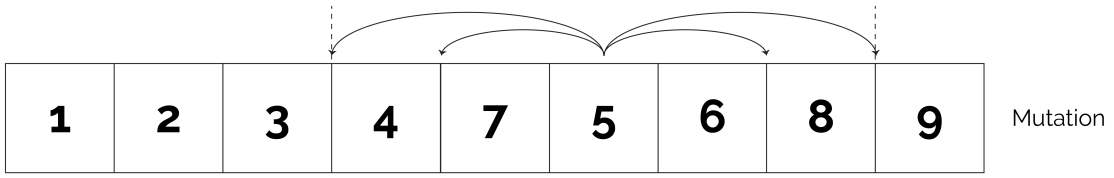


Figure 5.3: Example of the mutation operator

5.1.6. Selection

Elitism (and selection) is a third important step in a GA. Elitism (or survival of the fittest), is used to carry over the solutions with the best fitness value for the next generation. This is done to preserve the quality of the solutions to the next generations. As has been explained earlier, the newly formed population and the original population are pooled together. From this total population a certain number, $\mathcal{N}$, of individuals with the best fitness score are selected to be carried over to the next generation.

5.1.7. Decoding

The decoding of the chromosome is based on the information needed for the fitness function and the desired output, in this case which tasks are allocated to which mated station. The allocation of tasks should be deterministic, a sequence of tasks should always yield the same output, therefore no randomness can be involved. When allocating tasks to a workstation the assumptions as posed in Chapter 2 should be adhered to. For this purpose, a heuristic has been written to decode the chromosomes. The basis of the heuristics is based on three choices, the first being which operator

a task is assigned to, secondly if a task can be allocated to the current workstation and lastly, how many operators are assigned to each workstation.

First, the assignment of tasks to a mated station is done through the calculation of its earliest starting time on every available operator on the current workstation. The start time of a task depends on four factors: (1) the end time of the last allocated task to an operator, (2) a forward setup time, (3) idle time created by precedence relations caused by tasks allocated to other operators on the same workstation (I_1) and (4) idle time created by precedence relations and mounting positions by tasks allocated to operators on previous workstations (I_2). The start time for each model and operator is calculated and the minimum start time over all models is the start time used to decide to which operator the task is allocated to. Finally, the output is a list of operators in ascending order from earliest to latest start time. It is noted, that in case of a tie (i.e., the start times for 2 or more operators is equal), then the operator with the lowest index is chosen as best operator, as opposed to choosing one of the operators at random.

Secondly, a check has to be done to decide if a task can be allocated to the current workstation. The station time $t_{w,o,n}$ for each model is calculated, which includes all possible idle and setup times (τ and μ). In order to calculate setup times, the first and last (i.e., f and l respectively) allocated task to a workstation for each model must be tracked. As the backward setup time can vary between any combination of models (e.g., the setup from model n_1 to n_2 might be different than from n_1 to n_3 depending if a task is executed for a specific model), this time is averaged according to the first task that is relevant for a model, according to (4.31).

After all station times are calculated, they are averaged according to the model demand. If the average station time is lower that the takt time ($\mathcal{T}$) and the maximum station time is lower than the operators boundary ($\mathcal{T}_{max}$), then the task is allocated to the selected operator. Otherwise, the check is repeated for the next operator in the list. If the current task cannot be allocated to any operator, a new workstation is opened.

The choice of deciding the number of operators on each workstation is done through the efficiency (η) of each operator. First, a maximum amount of operators per workstation is set, O_{max}. This number typically depends on the available space on the assembly line. Next, a threshold value for the efficiency is set. The efficiency is the ratio between the useful time of an operator (i.e., time spent executing a task) and the takt time. This is in turn calculated for every model and is averaged according to the demand for each model. In other words:

$$\eta_{o,w} = \alpha_n \frac{\sum_{j \in (w,o)} \tau_j}{\mathcal{T}}. \tag{5.1}$$

After no more tasks can be allocated to the current workstation, the efficiency of each operator is calculated. If the efficiency of a single operator is lower than the threshold value, the number of operators for that workstation is reduced by one and the tasks are re-allocated to the remaining operators. This process is repeated until all operators reach the threshold value or until there is only a single operator left on the current workstation. The final output of the algorithm is the number of operators per workstation (O_w), number of workstations (W_{total}), a list of allocated tasks to each workstation and operator ($\mathcal{J}$) and the station times (t). The general heuristic can be seen in Algorithm 2.

5.1.8. Simple Lower Bound

An important factor in assessing the solution quality of the proposed heuristic is to compare the results to a lower bound, as the optimal solution is not known. The lower bound is calculated with a similar method as proposed by Scholl et al. (2013), however adapted to accompany the

Algorithm 2: Decoding Algorithm

Data: $\sigma, \mu, \tau, \pi, \mathcal{Z}, \mathcal{P}, \alpha, O_{max}, \mathcal{T}$
Result: $O_w, W_{total}, \mathcal{J}, t$
initialization: set $W_{total} = 1, O_w = O_{max}$,add σ_1 to $\mathcal{J}_{1,1}$, calculate $t_{1,1,n}$, add σ_1 to $f_{1,1,n}, l_{1,1,n}$ dependent on model;
$ii = 2$;
while $ii \leq noTasks$ **do**
  Calculate I_1;
  Calculate I_2;
  Decide best operator to assign task to based on earliest start time;
  for $o \in O$ **do**
    for $n \in N$ **do**
      Determine $\tau_{l,j,n}$ and $\mu_{j,f,n}$;
      Calculate $t_{w,o,n}$ based off idle and setup times;
    end
    if *mean station time* $\leq T$ *and max station time* $\leq T_{max}$ **then**
      add j to $\mathcal{J}_{w,o}$;
      Calculate $t_{w,o,n}, f_{w,o,n}, l_{w,o,n}$;
      break;
    else
      continue;
    end
  end
  if *task fits in mated station* **then**
    $ii = ii + 1$;
  else
    Calculate $\eta_{w,o}$ for all operators on current workstation;
    if *mean efficiency* $\geq \eta_t$ // $O_w = 1$ **then**
      $W_{total} = W_{total} + 1, O_w = O_{max}$;
      add σ_{ii} to $\mathcal{J}_{w,1}$, set $f_{w,1,n}, l_{w,1,n}$ dependent on model;
      Calculate $t_{w,o,n}$;
      $ii = ii + 1$
    else
      empty $\mathcal{J}_{w,o}$ and reset $f_{w,o,n}, l_{w,o,n}$ for all operators on current workstation;
      $ii = prev$;
      $O_w = O_w - 1$;
    end
  end
end

mixed-model nature of the ALBP.

The lower bound calculation is based off of a destructive improvement bound. First the capacity bound is calculated [1] (i.e., the lower bound assuming that there are no setup times):

$$O_{LB,n} = \frac{\sum_{j \in J} \tau_j}{\mathcal{T}}.$$

However, setup times should be included in the lower bound calculation. In order to do this, it is first recognized that in any given solution, a minimum of $J - O_{LB}$ forward setups are needed. The total time needed for forward setups is then $\pi_{total}^{J-O_{LB}}$, which is equal to the sum of the $J - O_{LB}$

[1] For an explanation of the variables used in this sections, see section 4.2

lowest setup times π_{j_1,j_2}, with $j_1 \in J$ and $j_2 \in S^{\tau}_{j_1}$. Here, $S^{\tau}_{j_1}$ is the set of possible forward setup times for task j_1, i.e.,:

$$S^{\tau}_{j_1} = \{J - (\mathcal{S}_{j_1} - \mathcal{S}^*_{j_1}) - \mathcal{P}^*_j - j_1\}.$$

The minimum amount of backward setup times is dependent on the maximum workstation length, β_w. For the SALBP, the amount of backward setups needed is equal to the amount of workstations, as each workstation has a backward setup. However, the mixed model nature of the problem can cause workstations to be empty for some models, while other models exceed the takt time. This can cause that there are more or less backward setup times than the amount of workstations.

In order to add backward setup times to the lower bound calculation, the minimum number of backward setups possible must be calculated. In any solution with a maximum takt time of $\beta_w \cdot \mathcal{T}$, the capacity bound becomes:

$$O_{\mu,n} = \frac{\sum_{j \in J} \tau_j}{\mathcal{T} \cdot \beta_w}.$$

Therefore, also a minimum amount of backward setups, equal to $O_{\mu,n}$, is needed for each model. The total amount of backward setup time needed is then equal to $\mu^{O_{\mu,n}}_{total}$, which is defined as the sum of the $O_{min,n}$ backward setups μ_{j_1,j_2}. Here, $j_1 \in J$ and $j_2 \in S^{\mu}_{j_1}$, where:

$$S^{\mu}_{j_1} = \{J - S^*_{j_1}\},$$

is the set of possible backward setups for a task j_1. The capacity available is $\bar{\mathcal{T}} = O_w \cdot \mathcal{T}$. Finally, the following problem must be solved for each model to calculate the lower bound:

$$\min\{O_{w,n} \geq O_{LB,n} | \sum_{j \in J} \tau_j + \pi^{J-O_w}_{total} + \mu^{O_{\mu,n}}_{total} \leq \bar{\mathcal{T}}\} \qquad O_w \geq O_{min}, \tag{5.2}$$

Then, the final lower bound is:

$$O_{min} = \sum_{n \in N} O_{w,n} \cdot \alpha_n$$

5.1.9. Design of Experiments

The algorithms performance depends on the chosen factors for the available parameters. In order to come to a correct choice for these parameters, a Design of Experiments is done. During DOE, multiple values of each parameter are checked in order to evaluate their influence on the results. In this case, two values for each parameter are chosen, a lower and upper bound value. The values of each parameter are chosen through an educated guess and can be seen in table 5.2.

Table 5.2: Table showing the lower and upper bound values for the parameters of the GA for the DOE.

Parameter	Lower Bound	Upper Bound
p_m	0.1	0.9
p_c	0.1	0.9
$\#gen$	10	100
$\mathcal{N}$	10	100
η_t	0.1	0.75
p_t	0.05	0.2

A 2-factor factorial design is performed, with an addition of 17 center points to check for possible curvature. This results in a total of $2^6 + 17 = 81$ runs. *StatGraphics18* is used to create the experimental design and to analyze the results. For each run, 10 samples are taken. A test case is created consisting of 90 tasks and 1000 models, taken from a real world example at VDL Nedcar. This ensures similar results to a an actual real-world case, however decreases the computation time.

Figure 5.4 show the main effects plot for each of the six parameters. A line is drawn from the lower bound to the upper bound (left to right), corresponding to the effect of the bound. As the goal is to minimize the cost function (fitness), which corresponds to the lowest point on the line. The longer the line, the more effect a certain parameter has. In addition, the steeper the slope of the line, the greater the magnitude of the main effect. A negative slope relates to a better result with a higher value of the parameter. It becomes clear that η has the largest influence on the results.

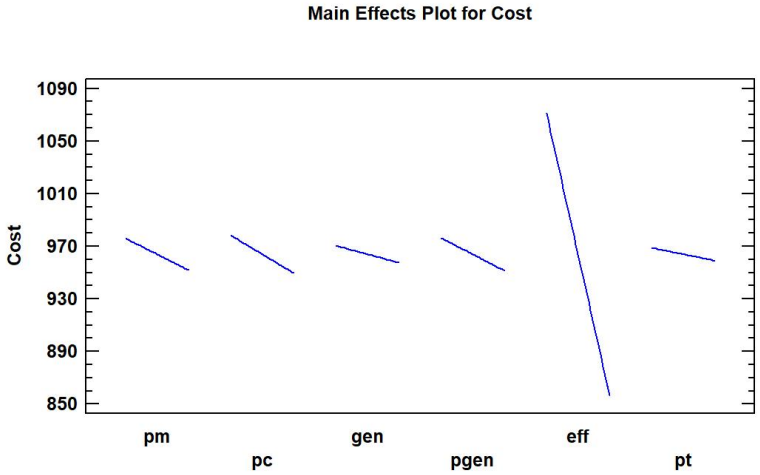


Figure 5.4: The main effects plot for the parameters of the GA. It can be seen that the threshold efficiency (eff) has the most effect, as it shows the steepest and longest line.

The main effects significance must also be checked, this is shown by the pareto chart in Figure 5.5. Here the standardized effects are shown of the main parameters as well as the two factor interaction between parameters. The blue vertical line represents the significance level. It can be seen that the threshold efficiency (η) and probability of crossover (p_c) have significant effects. Lastly, the optimal value for each parameter can be decided in *StatGraphics18*, this result is shown in Table 5.3.

Table 5.3: Optimal values of the DOE

Parameter	Optimal Value
p_m	0.9
p_c	0.9
$\#gen$	100
$\mathcal{N}$	100
η	0.8
p_t	0.2

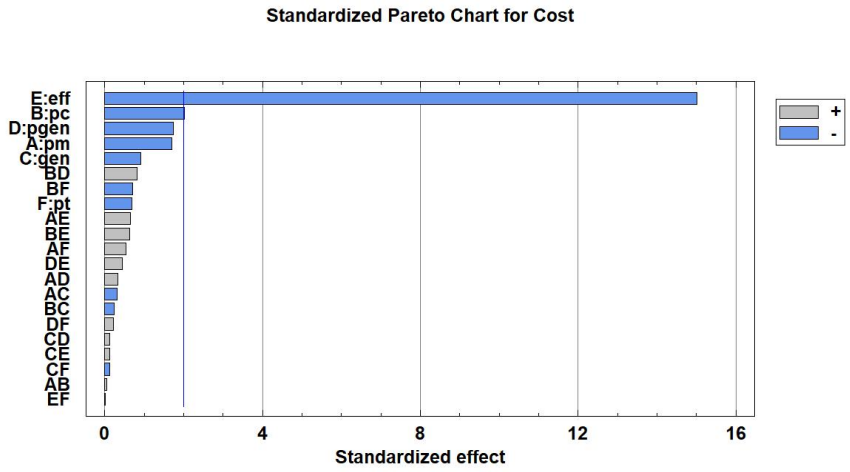


Figure 5.5: The standardized effect charts for the parameters of the GA. The threshold efficiency (eff) and probability of crossover (p_c) both are above the blue significance line. Therefore these 2 parameters have a significant effect.

5.1.10. Results

To compare the proposed algorithms to the MILP in Chapter 4, they were implemented into Matlab. The same test cases were used as in section 4.3.3. The rule-GRASP algorithm, used to create the initial population for the GA, produced the optimal solution consisting of 2 operators within less than 0.01 seconds. The GA will always improve solutions, therefore it can be concluded that the proposed algorithm can obtain similar results compared to the MILP in more reasonable time and can possibly be of use for real-world assembly line.

Table 5.4: Results of the MILP model for different test cases. The demand for each model was equal according to the number of models present in the test case. The time limit for each run was set to *7200* seconds.

Name	#jobs	#models	takt time	#operators	#workstations	objective	CPU time (s)
mertens	11	2	18	2	1	211.775	16.25
mertens	11	2	10	4	3	2433.19	16.2
jackson	13	2	18	3	2	322.525	1.2
roszieg	31	3	25	6	5	655.088	14.2

5.2. Summary

In this chapter a Genetic Algorithm was proposed as an alternative to the mathematical model as described in Chapter 4. The Genetic Algorithm uses mutation and two point crossover operators to generate new solutions for the ALBP. The initial population is created using the rule-GRASP algorithm in order to create a diverse and good population to start the GA with. Lastly, a decoding algorithm was given to translate the generated chromosomes into the desired solution (i.e., the amount of operators, workstations and the division of tasks among the operators). In addition, the algorithm showed to find the same answer to the test case that was also used in the previous chapter, in a shorter amount of time.

CHAPTER 6

Case Study

In this chapter a descriptions of the process and the results of applying automated line balancing at VDL Nedcar is described. For this purpose a single line section is chosen, referred to as Assembly Line X (ALX). In the first part of this chapter the process of collecting the data that is necessary top apply the algorithm as given in the previous chapter is described. In addition, an explanation of the procedure of reducing the amount of data that is collected is provided. Lastly, the results of applying the line balancing algorithm at VDL Nedcar are given.

6.1. Data collection and reduction

In order to implement automated line balancing at Nedcar, the correct data needs to be gathered, analyzed and adapted to fit in the previously described model. The main data that is needed is as follows:

- List of tasks with corresponding task time
- List of available mounting positions
- Precedence relations between tasks
- List of variants and their corresponding demands
- Forward and backward setup times
- Takt Time

In addition, some extra constraints may need to be added to the model depending on the assembly line section that is being balanced. Each assembly line is unique, e.g., some lines have tasks that have to be done on a specific workstation due to equipment or material constraints and other tasks might require two operators to work simultaneously on the same task. An important step to take is to analyze the assembly line thoroughly and take note of special constraints. It must however be noted that not each additional constraint should be adhered to strictly, as this may reduce the degrees of freedom in the eventual line balance. E.g., a task might require a specific piece of equipment that is only available at a single workstation, therefore a constraint could be added that fixes this specific task to that workstation. However, the cost reduction of the new line balance might outweigh the costs of moving the equipment. This only becomes apparent if the initial balance is done with the least amount of additional constraints.

An initial analysis of the data shows that ALX contains around 1000 tasks that have to be processed. Considering the amount of options that are installed on the assembly line, the total amount of variants is bounded by $\mathcal{O}(10^4)$ (an exact number is not known as not all option combinations can exist). It becomes apparent that the amount of variants quickly exceeds the amount of tasks, as the number of variants increases exponentially with an increasing amount of options. Therefore,

both the amount of tasks and variants have to be reduced in order to balance the line within a reasonable time frame.

Precedence relations are not directly available. The reason for this is that assembly line balancing is done by hand. The responsible engineer typically knows the order in which the tasks have to be done, however the exact relations are never written down. A similar principle applies to setup times. These are added later in the balance if it becomes apparent that an operator has to cover extra distance between tasks. The methods of obtaining both data types is explained in the next section.

6.1.1. Task reduction

As stated before, the total amount of tasks that needs to be allocated to ALX is around 1000. Balancing the line for all tasks separately is accurate, however it is very time consuming and not necessary due to "direct" precedence relations between certain tasks. These "direct" relations state that certain tasks have to be done consecutively (i.e., no other tasks can be done in between by the same operator). These tasks typically consist of grabbing a part, walking to the vehicle and mounting the part (other variations may also occur). Therefore, it can be useful to group the tasks together into a larger task, decreasing the total amount of tasks that are needed to balance the line.

In order to create larger tasks, some new definitions need to be established. A task is defined as the smallest operation that is carried out by an operator, such as grabbing a bolt and installing it in an exact location. Next, a process is a group of tasks describing the installation of a main part of the car. The task of installing a bolt is part of the process of installing the exhaust system. Lastly, a job is the group of tasks carried out by an operator. A job does not contain a whole process, but contains multiple parts of multiple processes. The last definition can be used to reduce the amount of tasks needed to balance the assembly line.

Figure 6.1 shows an example of how the amount of tasks are reduced by combining them into larger task packages. Sub-groups of tasks belonging to a single process are typically allocated to a single operator at a workstation. These sub-groups of tasks can be extracted from the current line balance or can be created intuitively if new tasks or processes have to be introduced into the line. E.g., tasks 1 and 2 of Process A have to be done in order and are both allocated to workstation 1 and operator 1. The sub-group consisting of tasks 1 and 2 are then combined into a task package labeled as Process A1. The processing times of task packages are the sum of the processing times of the individual tasks that form the package. Information regarding variants and mounting positions is also carried over. This leads to a reduction from 1000 tasks to 300 task packages that need to distributed among the line for ALX.

6.1.2. Variant Reduction

As stated before, the number of total variants in ALX is bounded by $\mathcal{O}(10^4)$. A high amount of variants can make the line balancing very accurate, as no work overload is created if a variant enters the line that has a very low demand. However, if a large amount of variants are considered, the problem changes into a task shop scheduling problem (i.e., scheduling tasks to an operator depending on the variant that is passing through the line at any given moment). Thus it is necessary to strongly reduce the amount of variants that are taken into account.

Each task contains certain variant information provided by the Process Development engineers. The information provided contains very specific and elaborate information of the process-variant a task is relevant to. A process-variant is defined as a sub-component of a single variant, see Figure 6.2 for more detail. Depending on the process a task is part of, variant information such as; model number, option number, driver side, engine specs etc. may be provided. As can be

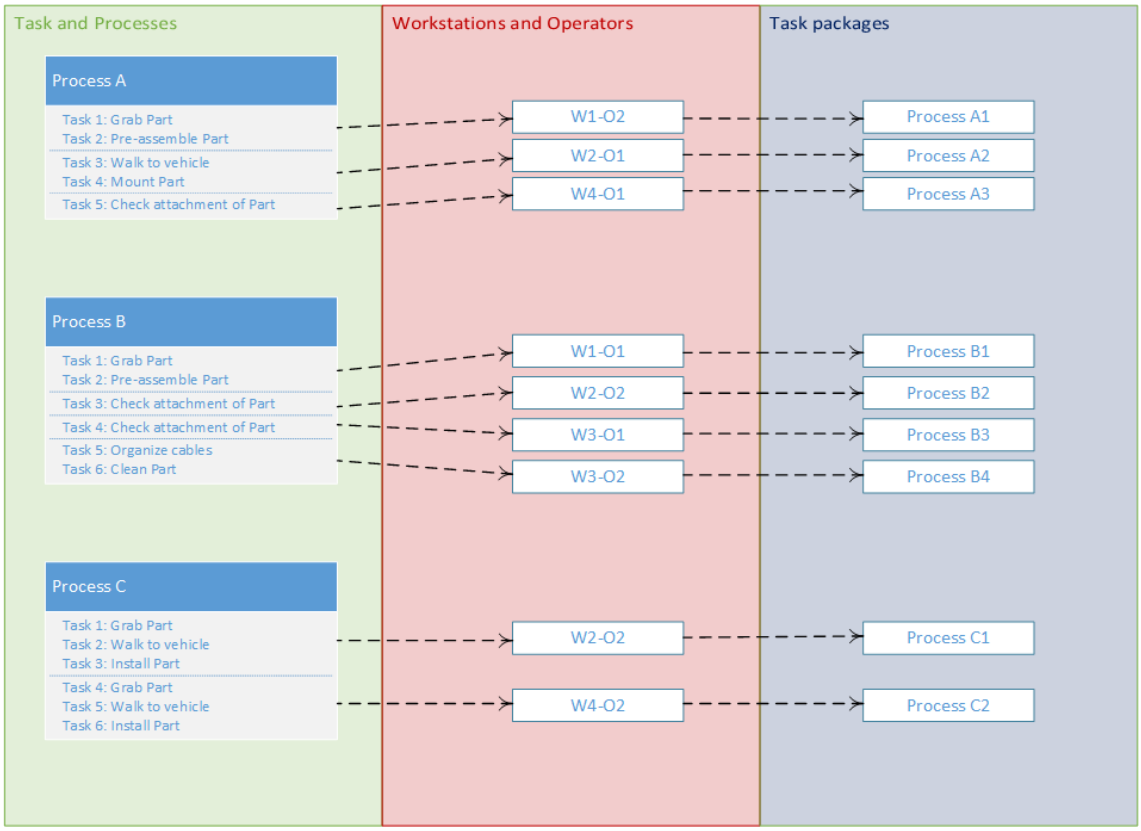


Figure 6.1: Diagram showing the reduction of tasks into task packages.

seen in Figure 6.2, variants contain all information regarding options. In contrast, process-variants contain information that is relevant to a specific task (or task package). E.g., a task that is done for Process Variant C2-03 would be carried out on Variants 1 and 2 on the assembly line, as variants 1 and 2 are of the correct model, driver side and both contain Option C.

The first step in reducing the number of variants is to check which combinations of process variants exist in the line. For this, a list of cars that have passed through the assembly line is used. This list consists of a car ID and the process-variants that are relevant to the car. A single car can consist of up to 500 process-variants. Each car is checked whether it contains one of the process-variants that occur on the assembly line, which results in an 0-1, $n \times p$ matrix, where n is the amount of cars and p the amount of process-variants. Each unique row of this matrix is then a single variant. This results in a reduction of around 90% when compared to the total amount of variants that can occur in the line. This method also returns the demand of each variant.

6.1.3. Generating Precedence

There are multiple ways to gather the information necessary to create the precedence graphs that are needed to balance the assembly lines. Klindworth et al. (2012) propose a way to approximate precedence graphs by using information regarding previous line balances. Previous line balances can find independencies between different tasks (i.e., if two tasks interchange position between two previous balances then no precedence relation can exist between those two tasks). They combine this information with known dependencies, gathered from CAD databases and experts with knowledge of the assembly line. Otto and Otto (2014) further extend this approach by considering the modular design used in the automotive industry. If one module has to be installed before another

	Model	Driver Side	Fuel	Country	Option A	Option B	Option C
Variant 1	M14	Right	Diesel	USA	No	Yes	Yes
Variant 2	M25	Right	Petrol	Japan	Yes	No	Yes
Variant 3	M38	Left	Electric	USA	No	No	No

	Model	Driver Side	Fuel	Country	Option A	Option B	Option C
Process Variant A1-01	M38	NA	Electric	USA	NA	NA	NA
Process Variant B3-03	NA	Left	Petrol	NA	NA	NA	NA
Process Variant C2-03	M14/ M25	Right	NA	NA	NA	NA	Yes

Figure 6.2: Example defining the difference between variants and process variants. The top table shows possible variants, where for each column it is specifically noted which options it is relevant for. In the bottom table example of process-variants are given. Here, only the information of a variant is given relevant to the task.

module, then all tasks regarding the first module precedes all the tasks of the second module. In addition, Otto and Otto (2014) also introduce novel ways to conduct interviews by stating the way questions regarding precedence relations should be asked and how to incorporate these into precedence graphs.

First, the method of generating precedence graphs by using the former method as described in the previous paragraph has been tested. Line balances over the course of one year were used. This did result in some independencies between tasks, however it was noticed that the overall line balance does not change enough. Usually, only small changes to the line balance are made and more vigorous changes that are needed for the generation of a precedence graph are rare. Tasks are also added and removed from the line continuously due to model updates or the re-assignment of tasks from another assembly line. This can restrict the amount of precedence relations gained from this method. However, it must be noted that results may vary between assembly lines. Some assembly lines are updated more often than others and in more extensive ways. It should therefore be tested whether this method can induce some relations.

Due to the fact that the precedence relations are not directly available, a method similar to the latter one described in the previous section is applied. The precedence graph is made through interviews with the responsible engineer who does the day to day line balancing and by walk through of the assembly line. A schematic overview of the generations is given in Figure 6.3. The basic concept consists of two phases; A coarse walk through of the precedence relations to filter out large groups and a finer walk through to filter out any remaining, individual relations.

Phase 1

An initial step in the generation is to find large independencies between tasks and groups of tasks (i.e., groups of tasks that do not have any precedence relations between them). Typically, in a multi-manned assembly line, tasks are duplicated over both sides of the vehicle, as a large portion of the vehicle is mirrored. Mirrored tasks that both relate to a similar part are therefore pre-

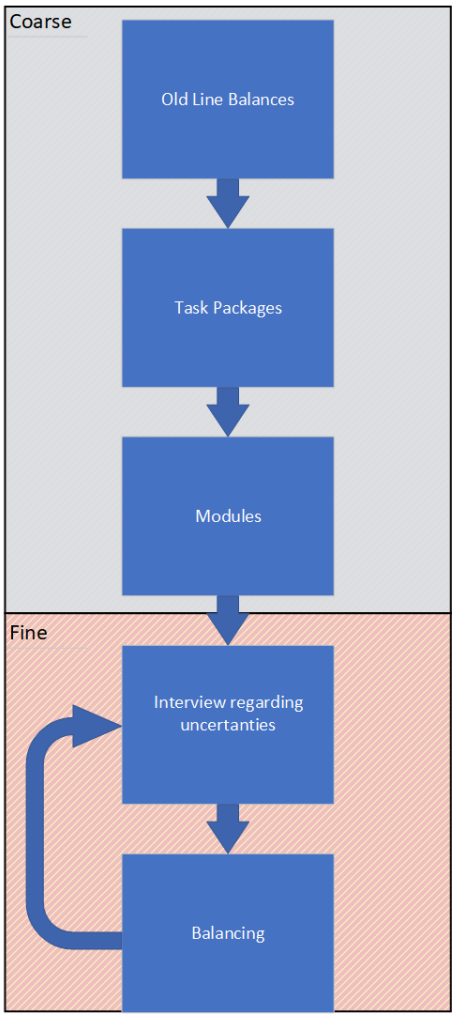


Figure 6.3: Overview of the steps that are taken to collect the precedence relations between the task packages. The top portion consists of the first phase consisting of coarsely collecting the relations and the bottom part of the last phase in which the relations are collected in a more detailed procedure.

confirmed independencies, and can be ignored during the initial phase. Next, relations between modules are extracted, as has been previously proposed by Otto and Otto (2014). This is done through interviews with the responsible engineer and walkthroughs of the line to visually see the position of certain parts. Lastly, relations between task groups are added. As previously explained in section 6.1.1, task groups are extracted based on the operator and workstation they are assigned to. These packages are assigned in order, as the total process has to be done in a specific order. Therefore, a precedence relation exists between task packages, where packages assigned on earlier workstations have to be done before packages assigned to later workstations. For example, in Figure 6.1 Process A1 has to be completed before Process A2 can be done. In this stage, task packages assigned to the same workstation but to different operators are assumed to not have any precedence relations.

Phase 2

Phase 2 consists of a much finer walkthrough of the precedence relations. Here, questions are asked as also stated by Otto and Otto (2014). Simple "yes/no" questions are asked, stating whether task A has to be done before tasks B or vice versa. This is not done for every task, but for relations

that were unclear from the prior phase. Then, the algorithm is run and task lists are generated. These are run through with the engineer and relations between tasks are checked. Even though this method does not extract all possible relations between tasks, it proved to be a useful method, as relations were more easily seen. This process is repeated multiple times, until most relations have been collected.

6.1.4. Setup Times

Lastly, the forward and backward setup times have to be calculated. This is done by defining setups between mounting positions, instead of defining them for each task pair individually, as this requires a high amount of information needed that is not readily available. Each task package is assigned a unique mounting position, according to the ones given in Figure 6.4. Some individual tasks already contain setup times (i.e., time needed to walk to grab equipment/material or to walk to the vehicle). As this is the case, and because it is difficult to find the exact time that has been added, it was decided to add a low amount of forward and backward setup times in order not to add more setup times than occur. Standard times between mounting position are added for both the forward and backward setup times, according to Table 6.1. This also includes and additional mounting position 12 that is not included in Figure 6.4. This mounting position is an off-line pre-assembly, typically done on the side of the assembly line.

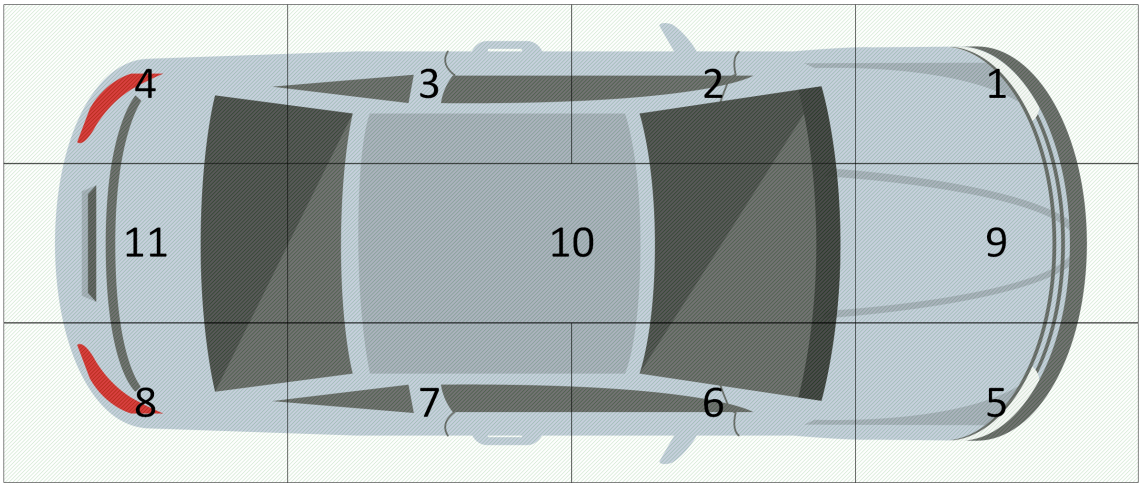


Figure 6.4: The mounting positions available to each task package.

6.1.5. Extra Constraints

As explained previously, every assembly line has additional extra constraints that must be included in the line balancing algorithm. These constraints can be related to material, equipment or operators. For ALX the following additional constraints have to be added to be kept track off:

Fixed Equipment : A large piece of equipment is used to fill the brake fluid. Moving this equipment is expensive, therefore the task package related to it is fixed to the workstation the equipment is positioned at (coupling process). Furthermore, the decoupling of the equipment has to take place a few workstation later, and, no tasks can be assigned to the workstations in between the coupling and decoupling of the equipment.

Multiple operators : The front end module (FEM) requires two operators working simultaneously. The two task packages related to this operation have to be carried out first in a station load at the workstation they are assigned to.

Table 6.1: Tables showing the forward and backward setup times between various mounting positions

(a) Forward setup times

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	1	2	3	2	3	4	5	1	2	4	1
2	1	0	1	2	3	4	5	3	2	1	2	1
3	2	1	0	1	4	5	4	3	3	1	2	1
4	3	2	1	0	5	4	3	2	4	2	1	1
5	2	3	4	5	0	1	2	3	1	2	4	1
6	3	4	5	4	1	0	1	2	2	1	3	1
7	4	5	4	3	2	1	0	1	3	1	2	1
8	5	3	3	2	3	2	1	0	4	2	1	1
9	1	2	3	4	1	2	3	4	0	3	5	1
10	2	1	1	2	2	1	1	2	3	0	4	1
11	4	2	2	1	4	3	2	1	5	4	0	1
12	1	1	1	1	1	1	1	1	1	1	1	0

(b) Backward setup times

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

For the first constraint a penalty function is added to the GA. If the task related to the coupling of the equipment is not assigned to the correct workstation, a large penalty is added to the fitness function. Next, if a mated station (w, o) contains the coupling task after it is closed, then the number of current workstations (W_{total} in algorithm 2) is increased by x instead of 1, with x being the number of empty workstations needed.

The second constraint is completely added to the decoding algorithm. If the current task is one of the two tasks related to the FEM, then a new workstation is opened, and both task packages are assigned to different operators on the newly opened workstation. It is also noted that the minimum number of operators on this workstation is equal to 2 instead of 1.

6.2. Results

In this section the results of implementing the algorithm as given in Chapter 5 and the data reduction as explained for ALX at VDL Nedcar is given. Due to confidentiality most of the results cannot be shown, however a short explanation is still provided.

6.2.1. Assembly line balancing

The most important results to consider are the efficiency of the line balance, the smoothness index and the lower bound. The overall line efficiency is the average of the individual efficiencies of all operators, in other words:

$$\eta = \frac{1}{O_{total}} \sum_{w \in W} \sum_{o \in O_w} \eta_{o,w},$$

where $\eta_{o,w}$ is as stated in (5.1). The smoothness index results directly from the horizontal balancing fitness functions as given in section 5.1.7, however scaled according to the takt time:

$$SI = \frac{\sum_{n \in N} \sum_{w \in W} \sum_{o \in O_w} |t_{w,o,n} - \mathcal{T}|}{\mathcal{T}}.$$

The lower bound calculation of section 5.1.8 resulted in an efficiency of 95%. An educated guess would therefore put the optimal solution around 90%, if all setup times and idle times are included. The following parameters were used to generate the new line balance according to the results from the DoE in section 5.1.9:

Table 6.2: Table showing the parameters used to balance ALX

Variable	Value
$\mathcal{T}_{max}$	$1.5 \cdot \mathcal{T}$
η_t	0.75
$\mathcal{N}$	100
O_{max}	3
p_c	0.9
p_m	0.9
p_t	0.2

The only parameter that was altered from the DoE results is the number of generations. It became apparent during testing that the fitness function still decreases after 100 generations, due to the horizontal balancing parameter. It was therefore chosen to swap out the limit on the number of generations by a time limit. This limit was set to *7200* seconds, as finding a better solution is of more importance than finding a solution quickly.

Overall, an increase from 78% to 85% was seen in the line efficiency, with less operators and workstations used. The smoothness index has been decreased from 10733 to 3498, thus resulting in a more even station load throughout the assembly line, which can be seen in Figure 6.5. The maximum station load has been decreased from 205% to 128%.

It must be noted that the constraint that multiple operators cannot be allocated to the same mounting position has been relaxed due to the size of the mounting positions. Only allowing a single operator did result in less operators, however the amount of workstation needed increased significantly. It was therefore chosen to allow 2 operators to perform tasks at the same mounting position. This is similar to the current line balance.

Lastly, the station times of the operators for the newly generated line balance were close to the takt time. Considering the addition of setup times and idle times, it may be possible to further increase the efficiency by $3-5\%$, however this might require some changes in the proposed model.

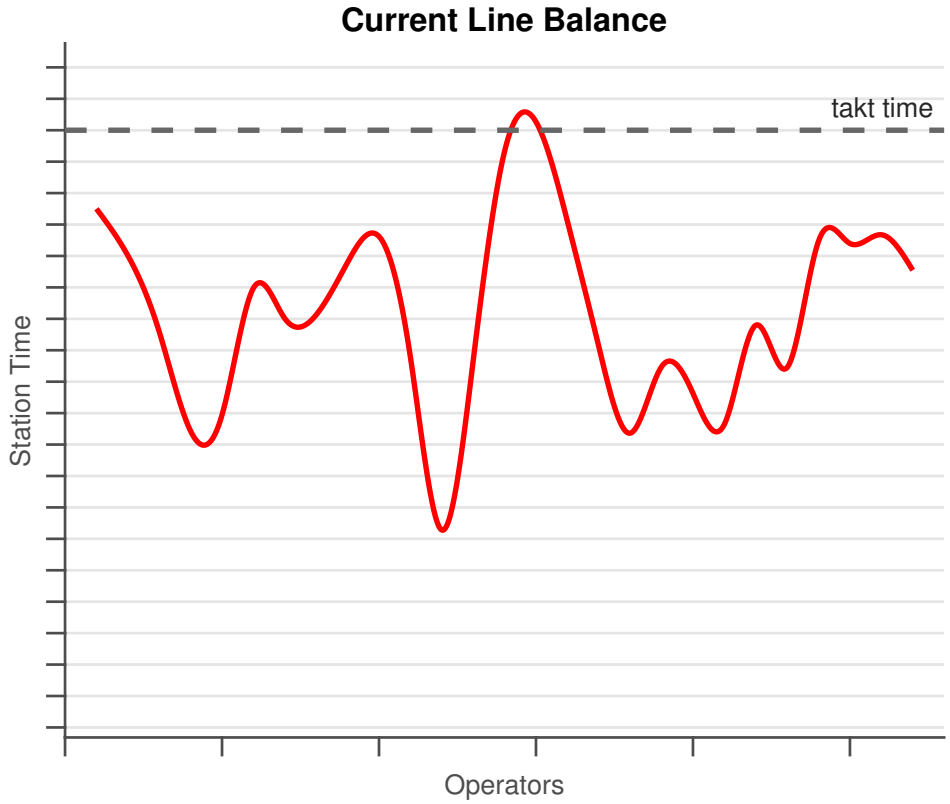
6.2.2. Assembly Line Sequencing

The car sequencing model as described in Chapter 4 was implemented into IBM ILOG CPLEX and a Python API. Car sequencing rules as given in section 2.3 were applied to the new line

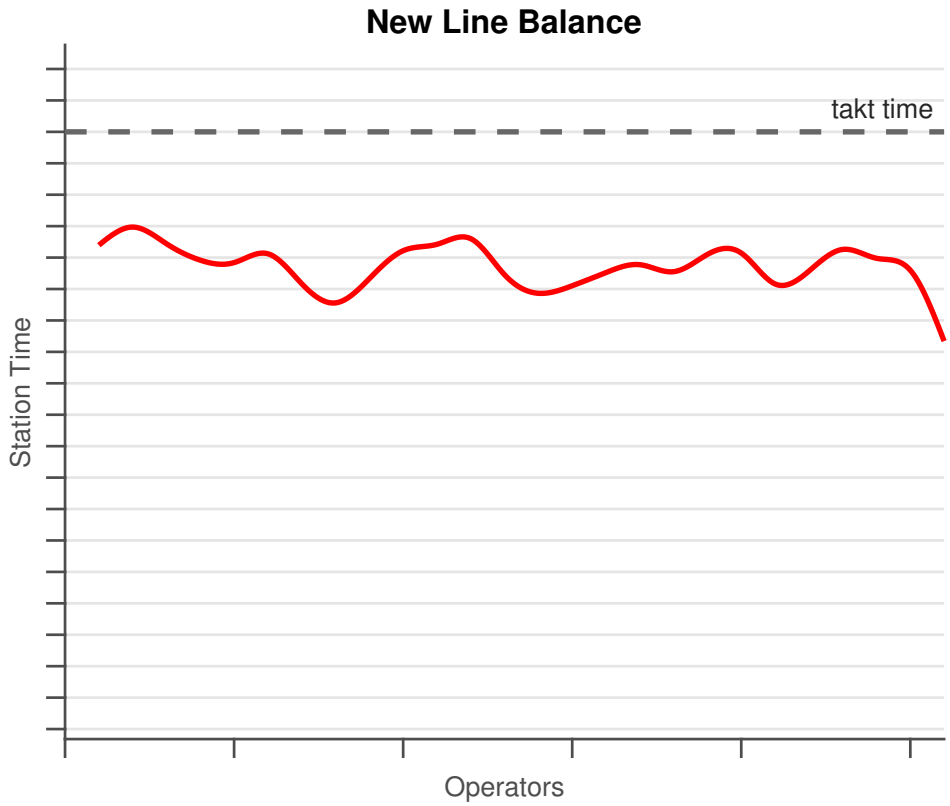
balance. However, the newly generated rules did not overrule the current rules used at VDL Nedcar. Therefore, the CS was tested for the current rules. This was done for six separate day packages (i.e., a package containing the cars that have to be produced during a certain period lasting one day). A total of 16 rules were implemented, leading to 16 different variants and 53 models. The model solved all 6 test cases in less than 600 seconds, all resulting in objective functions equal to 0 (i.e., no work overload was directly created). This is a reduction of 90% in computation time and 100% in work overload compared to the current situation.

6.3. Summary

In this chapter a case study was explained describing the methods used to implement the assembly line balancing models, as described in the previous chapters, at VDL Nedcar. A single line was chosen to test the model on. The number of tasks and variants were reduced in order to more easily solve the ALBP. Next, a method to gather precedence relations was described. This method is based on the current line balance and interviews with the responsible engineer. Lastly, additional constraints that describe the assembly line and the results of implementing the model were given. The results showed that the line efficiency was increased by 5% and that the workload was more even over the operators.



(a) Current line balance



(b) New line balance

Figure 6.5: Distribution of the station times over all operators for the current and new line balance. It can be seen that the newly generated line balance has a smoother overall workload between operators. In addition, the large peaks have been removed in the new balance.

CHAPTER 7

Conclusion and recommendations

7.1. Conclusion

This project focused on investigating the automation of assembly line balancing at VDL Nedcar. Most importantly, the following research question was stated:

How can the current assembly lines be balanced automatically so that, when changes are made to the assembly line (i.e., process or volume), the highest possible efficiency is achieved?

In order to answer this question, some subquestions were also stated that have been studied throughout this thesis. First, the current process of line balancing had to be understood in order to gain insight into how automating the process can help the current engineers, and what is needed in order to automate the process. Here it became apparent that line balancing is a day-to-day process, and that there are many reasons why a line has to be rebalanced. Furthermore, line balancing is a very time consuming job. Depending on the reason why a line has to be rebalanced, it may take a a significant amount of time to find a reasonable balance. This is due to the fact that line balancing is still done "by hand". It has to be checked manually whether a task can be assigned to an operator, if all models can be made and so on. The engineers use smart tricks in the way process are grouped and defined, and how they are ordered. These tricks should be used while automating the line balancing.

Next, the constraints that are needed to balance the line were sought out. It was found out that there are multiple constraints that can be included in the model, however not all of them have similar importance and can be left out. Most importantly, Mixed Models, Mounting positions, setup times (such as walking and tool changes) and multiple operators per workstation have to be accounted for, besides the standard constraints regarding precedence relations and takt time. Mixed model balancing was accounted for by relaxing the takt time constraints. Some models should be allowed to exceed the takt time, as long as on average the takt time is still met. This is in accordance with how the line is currently being balanced. Setup times were found to be dependent on the mounting positions of tasks. Different task sequences lead to large differences in the total amount of setup time that is added, and therefore lead to more or less operators that are used. Lastly, multiple operators per workstation have to be considered in order to be able to execute all the tasks necessary to assemble a car. However, care has to be taken while doing this as operators cannot occupy the same mounting positions simultaneously and waiting times may be incurred due to precedence relations.

Fluctuations in demand can be counteracted with horizontal balancing, i.e., minimizing the total difference between the station load of each model with the takt time. This assures that if slight differences occur in the demand of certain models, the line does not have to be rebelanced. In addition, the sequence in which models enter the assembly line also has to be taken into account. A line balance is only feasible if an operator is not overloaded with too much work. Models that

exceed the takt time can only occur considering specific rules that denote how many times they can occur consecutively. If these rules are not met then too much work overload is created and the line has to be halted.

In order to solve the ALBP, an MILP has been developed. However, even for small problems the model was not of practical use. A large amount of design variables and constraints is needed to define the model, thus for now it is not of practical use. For that reason, a GA has been developed that can solve the problem within reasonable time. In addition, an algorithm has also been developed that decodes the solutions generated by the GA. This heuristic assigns tasks to operators based on their overall efficiency and starting time.

Adjustments to the current data at Nedcar had to be made in order to apply the line balancing algorithm. The number of tasks and variants should be kept to a minimum, as these two variables greatly influence the effectiveness of the algorithm. However, these variables should not be reduced too much, as this in turn reduces the overall degrees of freedom of the line balance. The GA in combination with the proposed decoding heuristic proved to be successful in rebalancing an existing line, showing improvements of close to 6% in the overall efficiency. This is already a significant improvement for the line balance. This can further be improved considering the recommendations in the following section. In addition, multiple other constraints have been added to the model, to consider line specific restrictions.

Overall, it can be concluded that the proposed model can be used to balance real world automotive assembly lines and provides similar or better results in comparison to the current line balance. This is a big step, as previous literature, as of the time of writing this, has not included the same type of constraints and, the best of our knowledge, never been tested on a large automotive assembly line.

7.2. Recommendations

The following recommendations are made:

- The ALSP should be tested for more days to evaluate the full effect of the proposed model, to see if the optimum solution can always be achieved. A side-by-side test can be done with the current line sequencing software, to evaluate the exact differences and to see what is needed to implement the model.
- Mounting positions should be kept small enough so that only a single operator can perform a task at that position. If the mounting positions are too large then too many workstations are needed to perform all tasks. It is therefore recommended that around 24 unique mounting positions are allocated to the car.
- A more effective way of keeping track of precedence relations should be investigated. As this has proven to be a very time consuming process, more use should be made of CAD drawings and other schematics. Furthermore standard interview questions and a standard format for keeping track of the relations should be made. The process has to improve if these relations have to be gathered for the entire factory.
- Costs related to moving equipment, material and other components should be added to the model. This relaxes some fixed station assignment constraints that have been implemented. It can then be checked if the cost reduction related to an increase of the line efficiency outweighs the costs for the movement of equipment or materials. Relaxing these fixed constraints can possibly have a slight impact on the efficiency as results showed that there is room for a 3% efficiency gain.
- It should be investigated if some task packages can be made smaller as this can increase the efficiency of some workstations, while possible emptying other workstations. This can lead to

an overall decrease in both the amount of operators and workstations used, possible leading to the extra 3% efficiency gain.

Bibliography

Alghazi, A. and Kurz, M. E. 2018. Mixed model line balancing with parallel stations, zoning constraints, and ergonomics. *Constraints*, 23(1):123–153.

Álvarez-Miranda, E. and Pereira, J. 2019. On the complexity of assembly line balancing problems. *Computers and Operations Research*, 108:182–186.

Andrés, C., Miralles, C., and Pastor, R. 2006. Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, page 8196.

Battaïa, O. and Dolgui, A. 2013. A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics*, 142(2):259–277.

Baybars, l. 1986. A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science*, 32(8):909–932.

Becker, C. and Scholl, A. 2006. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694–715.

Becker, C. and Scholl, A. 2009. Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure. *European Journal of Operational Research*, 199(2):359–374.

Bolat, A. and Yano, C. A. 1992. Scheduling alogrithms to minimize utility work at a single station on a paced assembly line.

Boysen, N., Fliedner, M., and Scholl, A. 2007. A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674–693.

Boysen, N., Fliedner, M., and Scholl, A. 2008. Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 111(2):509–528.

Boysen, N., Fliedner, M., and Scholl, A. 2009. Assembly line balancing: Joint precedence graphs under high product variety. *IIE Transactions (Institute of Industrial Engineers)*, 41(3):183–193.

Bukchin, J., Dar-El, E. M., and Rubinovitz, J. 2002. Mixed model assembly line design in a make-to-order environment. *Computers & Industrial Engineering*, 41(4):405–421.

Chen, J. C., Chen, Y. Y., Chen, T. L., and Kuo, Y. H. 2019. Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in TFT–LCD module process. *Journal of Manufacturing Systems*, 52(May):86–99.

Emde, S., Boysen, N., and Scholl, A. 2010. Balancing mixed-model assembly lines: A computational evaluation of objectives to smoothen workload. *International Journal of Production Research*, 48(11):3173–3191.

Esmaeilbeigi, R., Naderi, B., and Charkhgard, P. 2016. New formulations for the setup assembly line balancing and scheduling problem. *OR Spectrum*, 38(2):493–518.

Falkenauer, E. 2005. Line balancing in the real world. *Proceedings of the International Conference on Product Lifecycle Management PLM*, 5:360–370.

Ford, H. and Crowther, S. 1922. *My Life and Work*. Doubleday.

Golle, U., Rothlauf, F., and Boysen, N. 2014. Car sequencing versus mixed-model sequencing: A computational study. *European Journal of Operational Research*, 237(1):50–61.

Hounshell, D. 1984. *From the American System to Mass Production, 1800-1932: The Development of Manufacturing Technology in the United States*. The John Hopkin University Press, Baltimore.

Kim, Y. K., Kim, Y., and Kim, Y. J. 2000. Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning and Control*, 11(1):44–53.

Kim, Y. K., Song, W. S., and Kim, J. H. 2009. A mathematical model and a genetic algorithm for two-sided assembly line balancing. *Computers and Operations Research*, 36(3):853–865.

Klindworth, H., Otto, C., and Scholl, A. 2012. On a learning precedence graph concept for the automotive industry. *European Journal of Operational Research*, 217(2):259–269.

Li, Z., Kucukkoc, I., and Nilakantan, J. M. 2017. Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. *Computers and Operations Research*, 84:146–161.

Martino, L. and Pastor, R. 2010. Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48(6):1787–1804.

Merengo, C., Nava, F., and Pozzetti, A. 1999. Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, 37(12):2835–2860.

Meyr, H. 2004. Supply chain planning in the german automotive industry. *Supply Chain Planning: Quantitative Decision Support and Advanced Planning Solutions*, pages 343–365.

Otto, C. and Otto, A. 2014. Multiple-source learning precedence graph concept for the automotive industry. *European Journal of Operational Research*, 234(1):253–265.

Sabuncuoglu, I., Erel, E., and Tanyer, M. 2000. Assembly line balancing using genetic algorithms. *Journal of Intelligent Manufacturing*, 11(3):295–310.

Salveson, M. 1955. The Assembly Line Balancing Problem. *The Journal of Industrial Engineering 6 (3),*, pages 18–25.

Scholl, A. 1999. Armin Scholl - Balancing and Sequencing of Assembly Lines. pages XVI, 318.

Scholl, A., Boysen, N., and Fliedner, M. 2013. The assembly line balancing and scheduling problem with sequence-dependent setup times: Problem extension, model formulation and efficient heuristics. *OR Spectrum*, 35(1):291–320.

Sivasankaran, P. and Shahabudeen, P. 2014. Literature review of assembly line balancing problems. *International Journal of Advanced Manufacturing Technology*, 73(9-12):1665–1694.

Tasan, S. O. and Tunali, S. 2008. A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing*, 19(1):49–69.

Tiacci, L. 2015. Coupling a genetic algorithm approach and a discrete event simulator to design mixed-model un-paced assembly lines with parallel workstations and stochastic task times. *International Journal of Production Economics*, 159(October):319–333.

Wee, T. S. and Magazine, M. J. 1982. Assembly line balancing as generalized bin packing. *Operations Research Letters*.

Yang, C., Gao, J., and Sun, L. 2013. A multi-objective genetic algorithm for mixed-model assembly line rebalancing. *Computers and Industrial Engineering*, 65(1):109–116.

Yang, W. and Cheng, W. 2019. Modelling and solving mixed-model two-sided assembly line balancing problem with sequence- dependent setup time. *International Journal of Production Research*, 0(0):1–22.

TU/e Technische Universiteit Eindhoven University of Technology

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

18-03-2020

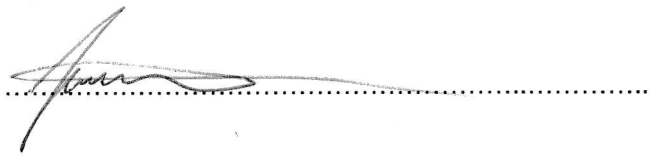
Name

J.B.H.C. Didden

ID-number

0881512

Signature



Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU