OPTIMIZATION OF THE ASSEMBLY SCHEDULING AT ROYAL TERBERG GROUP

SOLVING A MULTI-OBJECTIVE FLOW SHOP SCHEDULING PROBLEM WITH SUBSIDIARY-SPECIFIC RESTRICTIONS

Ruben Hampsink
Industrial Engineering & Management

October 2023

OPTIMIZATION OF THE ASSEMBLY SCHEDULING AT ROYAL TERBERG GROUP

SOLVING A MULTI-OBJECTIVE FLOW SHOP SCHEDULING PROBLEM WITH SUBSIDIARY SPECIFIC RESTRICTIONS

Master Thesis
Industrial Engineering & Management
University of Twente

October 2023

Author
Ruben Hampsink
MSc Industrial Engineering & Management

Royal Terberg Group B.V.
Newtonstraat 2
3401 AJ IJsselstein
The Netherlands

Supervisors
Wout op ‘t Ende
Data Science Consultant
Chris Louman
Data and Development Manager

University of Twente
Drienerlolaan 5
7522 NB Enschede
The Netherlands

Supervisors
Dr.ir. J.M.J. Schutten
1st supervisor
Dr. E. Topan
2nd supervisor

Acknowledgements

This thesis marks the end of my journey at the University of Twente, where I successfully completed both the bachelor's and master's programs. Throughout my time as a student, I had the privilege of acquiring valuable knowledge and achieving numerous accomplishments.

First, I would like to thank Royal Terberg Group B.V. for providing me with this interesting graduation opportunity. It has been an honour to meet and work with all the people within Terberg and its subsidiaries. My colleagues at Terberg were consistently supportive and genuinely interested in my project. They were eager to think along with me to overcome challenges. I felt that the project was taken seriously and it interesting to see that the outcome of this research serves as a base for their new scheduling tool.

I reserve special thanks for Wout op 't Ende, my supervisor at Terberg. Wout was willing to provide assistance and support throughout my entire graduation period. His invaluable guidance improved the quality of my experience and research at Terberg.

I would like to express my gratitude to my supervisors at the University of Twente, Marco Schutten and Engin Topan. Their expertise and guidance were invaluable during the graduation project. Their insightful feedback helped to refine my writing and clarifying my ideas throughout the process.

Last, I would like to thank my family and friend for their encouragement and unconditional support during my academic journey time and graduation period.

I hope you enjoy reading this thesis!

Ruben Hampsink
Oldenzaal, October 2023

Management Summary

The Royal Terberg Group assembles special vehicles and vehicle systems in subsidiaries all over the world. This research centres on 4 assembly lines. The other assembly lines are comparable to 1 of these 4 lines. The Process Owner Production of Terberg indicate that the assembly performance is not as desired. The current scheduling approach result in many penalties due to late delivery. Furthermore, the existing scheduling approach fails to effectively align and balance the assembly processes, consequently leading to overtime and idleness.

At the moment, the planner at each subsidiary makes the assembly plan based on experience and gut feeling. The planners have difficulties with creating a schedule that considers the variations, uncertainties and restrictions that are present at the subsidiaries. Therefore, the main research question is:

"What scheduling approach can be adopted by the assembly subsidiaries of Royal Terberg Group to effectively enhance the quality of the schedule, while considering the variations, uncertainties, and restrictions?

An analysis of the current situation indicates the similarities and differences between the assembly lines within the scope of the research. Terberg deals with variation in due dates, assembly times and required parts. All assembly subsidiaries of Terberg increasingly deal with the unavailability of parts. The consequences of these shortages have varying impacts on the (post)-assembly process. Moreover, each assembly line has restrictions related to, e.g., workstation-restricted jobs, sequence restricted jobs, synchronized cycle time movement, setup times and drying time.

The planning and scheduling process has two phases: the long-term and short-term planning and scheduling. In the long-term planning, the planner plans the jobs in a certain week. In the short-term scheduling, the planner determines the exact sequence of jobs within the week. Our research focuses in on the short-term scheduling, determining the best possible sequence of jobs in each stage and on each workstation.

We conduct a literature review to obtain modelling techniques for our scheduling problem. Based on the review and our insight, we propose solution approach configurations. A solution approach configuration consists of a constructive heuristic, improvement heuristic and a neighbourhood structure. The solution approach configurations generate the encoded solution. This is the sequence of jobs in each stage and on each workstation. We utilize a decoding algorithm to calculate the start- and end times and to track the inventory levels over time. We use the decode solution to calculate the objectives.

We perform experiments to determine the most promising solution approach configuration per assembly line. We propose a total of 20 solution approach configurations and select the appropriate parameters per solution approach and per assembly line. In the selection of the parameter values we make a trade-off between computational time and the quality of the schedule. We employ the best solution approach configuration to experiment with various modelling alternatives. These modelling alternatives are:

- Construct schedules with a simplified model that excludes the inventory levels;
- Determine the scalability of the solution approach configuration when applied to a monthly scheduling period;
- Compare the schedules with the schedule create by the planner in a real-life case.

We conclude that for all assembly line it is beneficial to exclude the inventory availability of the model during the construction of the schedule. The initial model updates the inventory levels over time. The new model excludes the updating procedure. The scheduling objectives differ minimal when we use the normal model after the construction of the schedule is done. The exclusion reduces the computational time with more than 50%. After we employ the improvement heuristic there is no difference between both models. The exclusion reduces the total computational time with 41.68%, 24.34% and 40.81% for the assembly line 1, 2, 3, respectively.

In the future, Terberg aims to increase the scheduling period to a month. Therefore, the scalability of the best solution approach configuration is important. We conclude that it is beneficial to update the parameters of the assembly line 1. The number of on-time finished jobs with no shortages increases with 43.25%. There is minimal improvement for assembly line 1 and 2. We conclude that the problem instance becomes too large too effectively improve schedule in a reasonable time. When we increase the instance size the number of schedule positions increases in each stage. A situation occurs where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

We conclude that the proposed solution approach configurations improve the quality of the schedule at each subsidiary. The KPIs show notable improvements when compared to a schedule created by the planner in a real-life case of each subsidiary. However, there are differences in these improvements when applying the proposed model to individual weeks versus a three-week combined approach.

At assembly line 1, the number of on-time jobs with no shortages increases with 20.69% when we apply proposed scheduling approach to 3 individual weeks. This number further increases to 75.86% when scheduling all three weeks combined with the proposed approach. In this case, the proposed approach schedules jobs in different weeks. This suggests that the initial job distribution across the weeks is suboptimal, and an extended scheduling period yields superior results. We conclude that the proposed approach is scalable, and it results in an even higher number of on-time jobs with no shortage.

At assembly line 2, we see an improvement of the makespan and the number of on-time jobs with no shortages. The makespan decreases both when we apply the proposed approach to the weeks individually and combined. The number of on-time jobs with no shortages is equal when we apply the proposed approach individually to the 3 weeks or combined. We conclude that the long-term planning is correct in this real-life case of assembly line 2. We conclude that in this case the proposed approach is scalable since the improvement is similar in both the individual and combined optimization. Nonetheless, the problem size becomes relatively large when aggregate the 3 weeks. This leads to a situation where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

At assembly line 3, the total makespan decreases with 6.05% when we apply the proposed scheduling approach is applied to the 3 individual weeks. However, the makespan only decreases with 1.97% when we apply the proposed approach to the 3 weeks combined. The increasing number of on-time jobs with no shortages is equal when we apply the proposed approach individually to the 3 weeks or combined. We conclude that in this case the proposed approach is not scalable. The problem size becomes extremely large. This also leads to a situation where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

The experiments did not include assembly line 4. However, we can still draw conclusion based on the conclusions of the other assembly line. Assembly line 4 shows resemblance with assembly line 1. Therefore, we conclude that same solution approach is applicable to assembly line 4.

All in all, the proposed solution approach generates schedules that consider the:

- Variation in due dates, assembly times and required parts;
- Uncertainty in availability of materials;
- Assembly line specific restrictions.

Additionally, we improve the (re)scheduling time. The developed model together with solution approaches significantly reduce the (re)scheduling time for the planner of each subsidiary. Additionally, we recommend Terberg to:

- Implement the solution approach per assembly line to create assembly schedules;
- Exclude the inventory levels in the construction of the schedule since the constructive heuristics already consider the shortages per job;
- Be careful with scaling up the scheduling period since the problem size of the assembly line 2 and 3 may become too large;
- Start experimenting with the proposed solution approach for assembly line 4 once they finalize and verify their assembly data;
- Continue tracking and establishing relationships between assembly times and required parts. This results in even more accurate schedules.

Table of Contents

1. Introduction

This chapter introduces the research performed at the Royal Terberg Group located in IJsselstein. Section 1.1 introduces the Royal Terberg Group and describes the divisions and departments related to the research. Next, Section 1.2 discusses the problem identification. Section 1.3 presents the research approach.

1.1 Royal Terberg Group

For over 150 years, the family owned Royal Terberg Group develops, assembles, and modifies special transport vehicles for all kinds of sectors. With 34 operating subsidiary companies in 14 countries, Terberg is a multinational supplier of (special) vehicles and *vehicle systems*. A vehicle system is an assembly of devices combined to perform one or more specific functions in a vehicle. Currently, Terberg consists of 4 divisions:

- Special Vehicles: Producer of terminal tractors for ports, distribution centres, heavy industry, shunting yards, and airports;
- Environmental Equipment: Producer of vehicles and vehicle systems for the waste management and recycling industry e.g., refuse collection vehicles;
- Truck Modification: Specialist in customer-focused vehicle modifications and solutions;
- Truck Mounted Forklifts: Producer of truck-mounted forklifts.

Figure 1-1 shows the corporate structure of Terberg including the divisions and subsidiaries. This research focuses on 3 assembly facilities: Terberg Benschop B.V., Terberg Machines B.V. and Dennis Eagle LTD. These subsidiaries are in the Special Vehicles and Environmental Equipment divisions. We focus on these 3 subsidiaries because the assembly lines at these locations have different layouts and restrictions. The layout of the other assembly subsidiaries of Terberg looks like one of the lines within the scope.

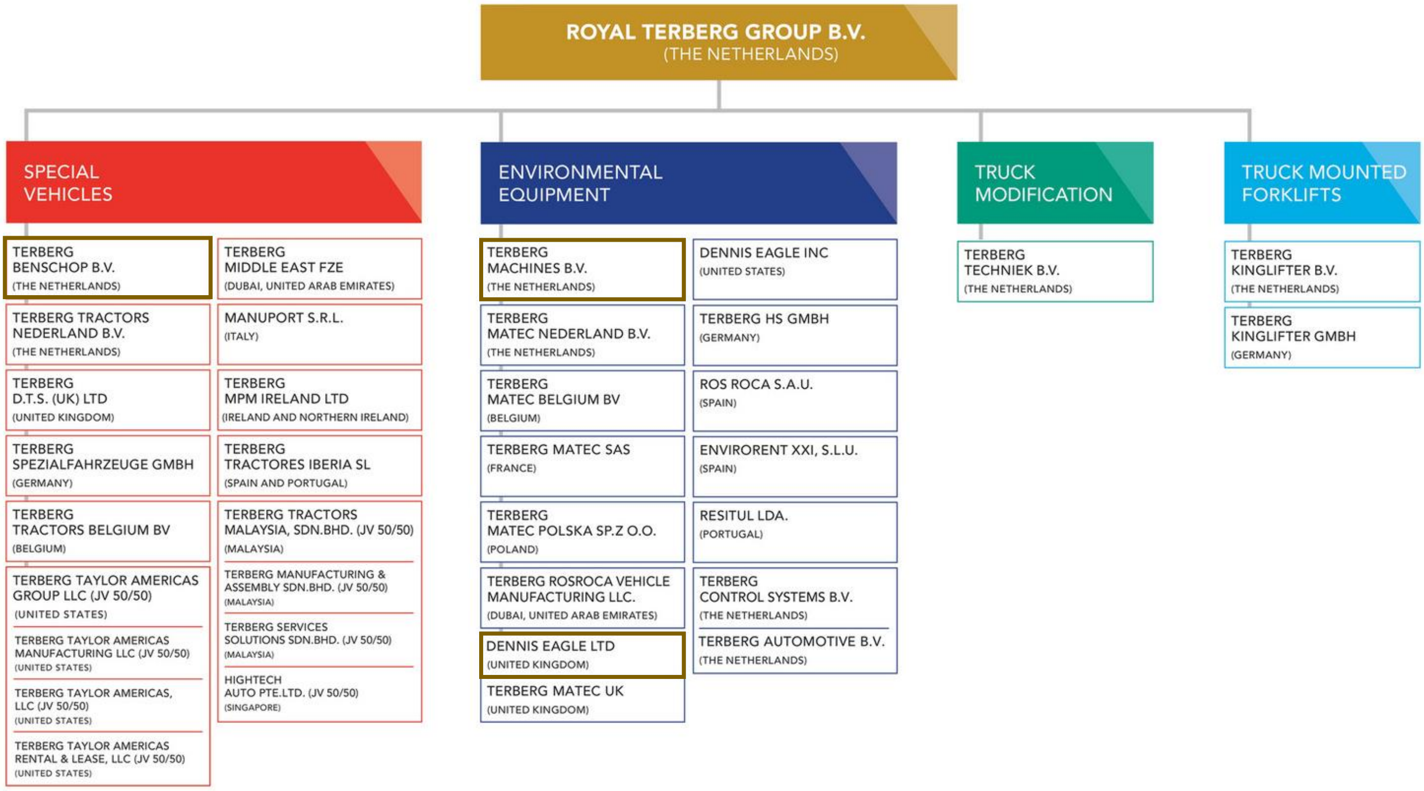

ROYAL TERBERG GROUP B.V. (THE NETHERLANDS)
SPECIAL VEHICLES
TERBERG BENSCHOP B.V. (THE NETHERLANDS)
TERBERG MIDDLE EAST FZE (DUBAI, UNITED ARAB EMIRATES)
TERBERG TRACTORS NEDERLAND B.V. (THE NETHERLANDS)
MANUPORT S.R.L. (ITALY)
TERBERG D.T.S. (UK) LTD (UNITED KINGDOM)
TERBERG MPM IRELAND LTD (IRELAND AND NORTHERN IRELAND)
TERBERG SPEZIALFAHRZEUGE GMBH (GERMANY)
TERBERG TRACTORES IBERIA SL (SPAIN AND PORTUGAL)
TERBERG TRACTORS BELGIUM BV (BELGIUM)
TERBERG TRACTORS MALAYSIA, SDN.BHD. (JV 50/50) (MALAYSIA)
TERBERG TAYLOR AMERICAS GROUP LLC (JV 50/50) (UNITED STATES)
TERBERG MANUFACTURING & ASSEMBLY SDN.BHD. (JV 50/50) (MALAYSIA)
TERBERG TAYLOR AMERICAS MANUFACTURING LLC (JV 50/50) (UNITED STATES)
TERBERG SERVICES SOLUTIONS SDN.BHD. (JV 50/50) (MALAYSIA)
TERBERG TAYLOR AMERICAS, LLC (JV 50/50) (UNITED STATES)
HIGHTECH AUTO PTE.LTD. (JV 50/50) (SINGAPORE)
TERBERG TAYLOR AMERICAS RENTAL & LEASE, LLC (JV 50/50) (UNITED STATES)
ENVIRONMENTAL EQUIPMENT
TERBERG MACHINES B.V. (THE NETHERLANDS)
DENNIS EAGLE INC (UNITED STATES)
TERBERG MATEC NEDERLAND B.V. (THE NETHERLANDS)
TERBERG HS GMBH (GERMANY)
TERBERG MATEC BELGIUM BV (BELGIUM)
ROS ROCA S.A.U. (SPAIN)
TERBERG MATEC SAS (FRANCE)
ENVIRORENT XXI, S.L.U. (SPAIN)
TERBERG MATEC POLSKA SP.Z O.O. (POLAND)
RESITUL LDA. (PORTUGAL)
TERBERG ROSROCA VEHICLE MANUFACTURING LLC. (DUBAI, UNITED ARAB EMIRATES)
TERBERG CONTROL SYSTEMS B.V. (THE NETHERLANDS)
DENNIS EAGLE LTD (UNITED KINGDOM)
TERBERG AUTOMOTIVE B.V. (THE NETHERLANDS)
TERBERG MATEC UK (UNITED KINGDOM)
TRUCK MODIFICATION
TERBERG TECHNIEK B.V. (THE NETHERLANDS)
TRUCK MOUNTED FORKLIFTS
TERBERG KINGLIFTER B.V. (THE NETHERLANDS)
TERBERG KINGLIFTER GMBH (GERMANY)

Figure 1-1. Corporate Structure of Royal Terberg Group

1.1.1 Special Vehicles Division

The Terberg Special Vehicles division assembles tractors and body carriers for different sectors. Customers use these vehicles to transport containers at airports, ports, distribution facilities or for transportation at other locations. We focus on the Terberg Benschop B.V. subsidiary of the Special Vehicle division. Figure 1-2 shows an example of a terminal tractor and a body carrier produced at Terberg Benschop B.V. This subsidiary offers a large variation in vehicles and additional options. Each product and option have different production times.

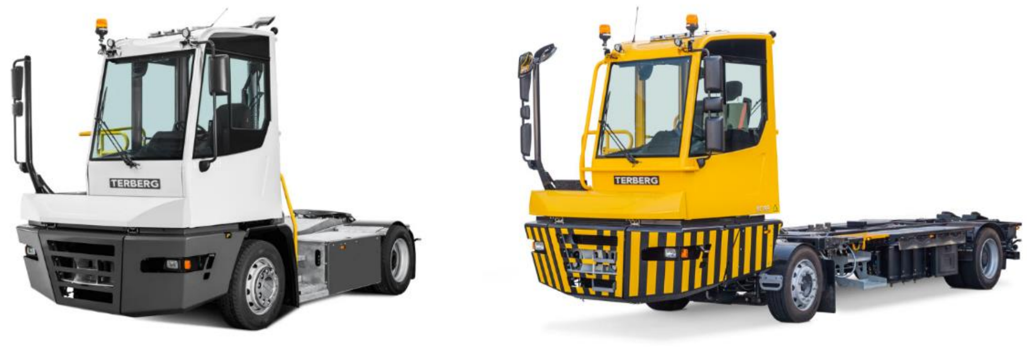


Figure 1-2. Terminal Tractor and Body Carrier of Terberg Benschop B.V.

1.1.2 Environmental Equipment Division

The Terberg Environmental Equipment division assembles vehicles for the waste management and recycling industry. We focus on Dennis Eagle LTD and Terberg Machines B.V. Both are subsidiaries of the Environment Equipment division. The Dennis Eagle LTD subsidiary is in Warwick, England.

Figure 1-3 shows an example of a bin lifter and front loader that Dennis Eagle LTD produces.

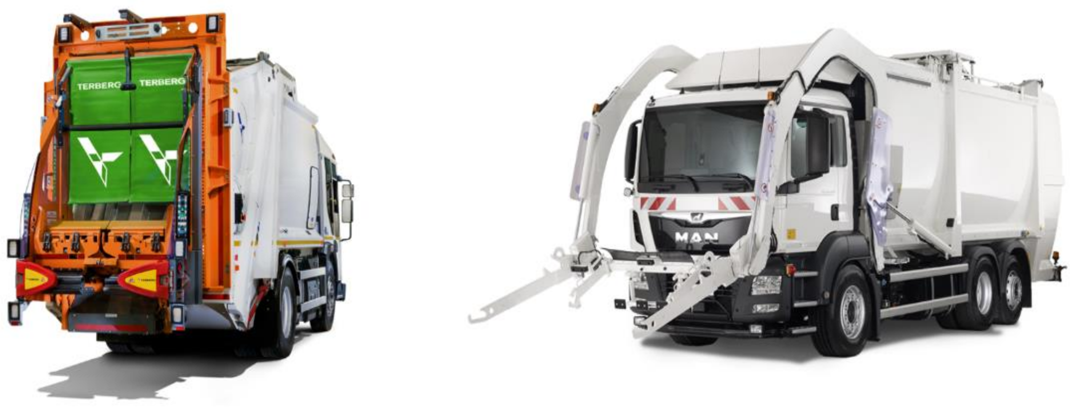


Figure 1-3. Bin Lifter and Front Loader of Terberg Dennis Eagle LTD

The Terberg Machines B.V. subsidiary is in IJsselstein. This subsidiary assembles all kinds of different bin lift systems for bin lifters. This is the orange part of the bin lifter in

Figure 1-3. Their main customer is the Dennis Eagle LTD and the Ros Roca S.A.U. subsidiary. The Ros Roca S.A.U. subsidiary is in Spain and has a similar assembly process as Dennis Eagle LTD.

1.1.3 Data Science Department

The research takes place within the Data Science Department. This department is part of the umbrella organisation Royal Terberg Group. In the Data Science Department, the employees use statistical methods, machine learning, algorithms, and other tools to analyse data and create predictive models. The Data Science Consultants create specific models for individual subsidiaries or generic models that multiple subsidiaries of the Royal Terberg Group use. Projects that they are currently working on are, e.g., a product tracking system, a product management system to sign off tasks and a planning and scheduling system.

1.2 Problem Identification

This section describes the problem identification. We use the Managerial Problem-Solving Method (MPSM) of Heerkens and Van Winden (2012) as a guideline to identify the existing problems at Terberg with a systematic approach. Section 1.2.1 presents the motivation to execute the research and presents the cause-effect relationships between problems in a problem cluster. Section 1.2.2 describes the relevant existing problems and the selection of the core problem. It also provides the scope and objective of the research.

1.2.1 Research Motivation

The Royal Terberg Group develops and maintains solutions for its subsidiaries. One of the areas where Terberg puts effort into is the improvement of assembly planning and scheduling. Data Scientists of Terberg indicate that the assembly performance is not as desired. Currently, the planner at each subsidiary makes the schedule based on experience and gut feeling. Terberg believes that this current way of planning and scheduling is not the most efficient and is outdated. This results in inefficient and/or infeasible schedules. Terberg looks for a planning and scheduling approach that deals with the present variation in due dates, assembly time and needed parts per product. Besides, in needs to consider the increasing number of uncertainties in terms of resources. Additionally, Terberg aims to increase the planning horizon. The goal is to create monthly schedules instead of weekly schedules. A longer-term schedule provides a higher-level view of tasks, deadlines, and resource allocations.

Figure 1-4 shows a problem cluster containing all relevant problems related to the assembly and the planning and scheduling process at Terberg. It contains the problems denoted by the employees within different departments at Terberg. It depicts the cause-and-effect relationships of the existing problems and brings structure to the problem context.

There are 3 main effects visible in the assembly subsidiaries at Terberg. First, more often than necessary the assembly stations are idle, or the assembly staff must work overtime (12). Second, the Data Science Consultant of Royal Terberg (2022) indicate that too much money is spent on penalties (17). These penalties occur due to a too-low percentage of on-time deliveries (15). Third, the planner at all subsidiaries experiences high workloads (16). The planner modifies the assembly schedule when the situation changes (14).

An assembly schedule that does not meet the desired assembly times and volumes causes these main effects (11). This applies to all the assembly subsidiaries of Terberg. The misfit originates from the fact that there are too many uncertainties and variations that influence the assembly processes (8). The current assembly planning and scheduling approach is not proactive and cannot deal with these uncertainties and variations (10).

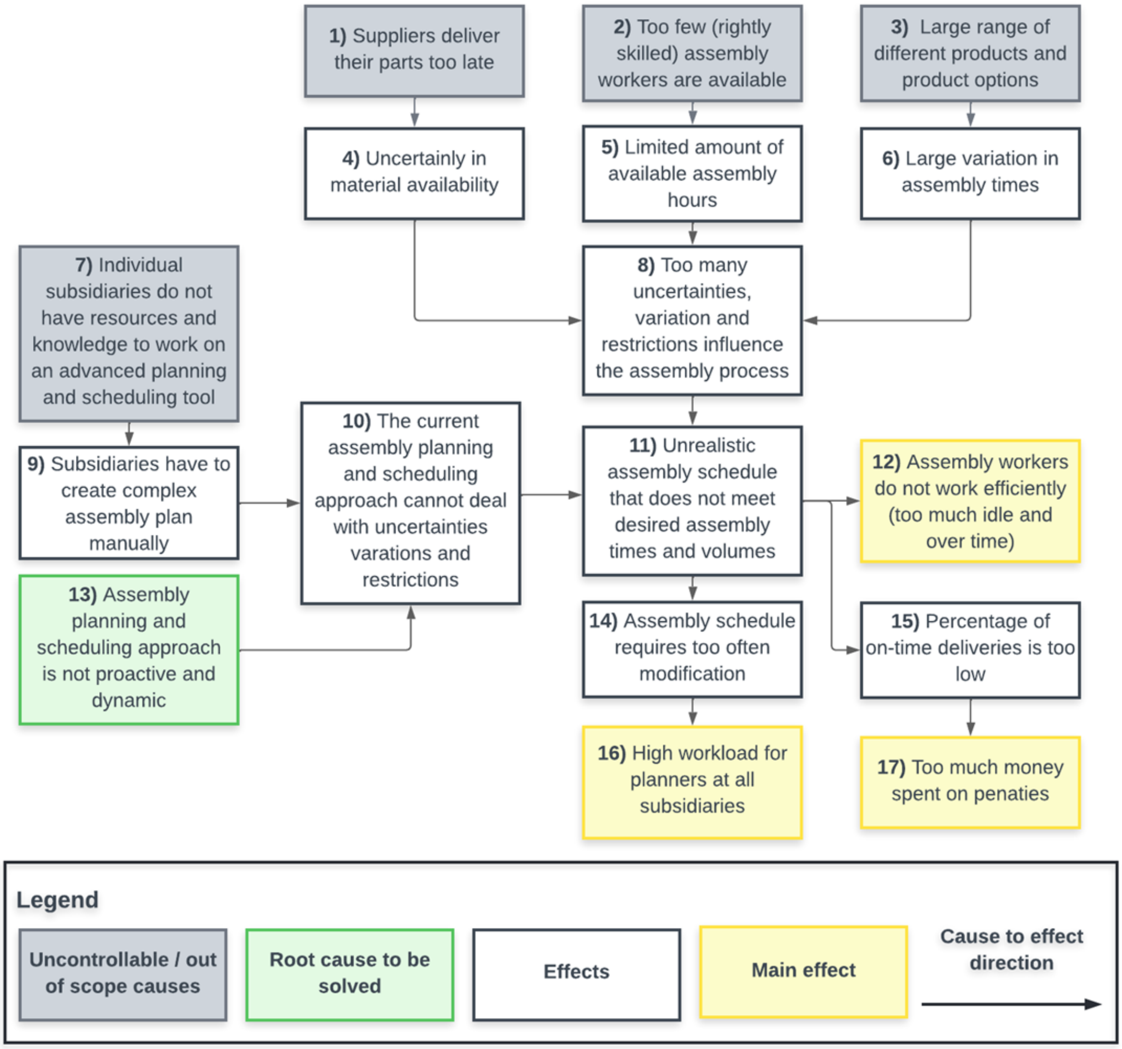

1) Suppliers deliver their parts too late
2) Too few (rightly skilled) assembly workers are available
3) Large range of different products and product options
4) Uncertainly in material availability
5) Limited amount of available assembly hours
6) Large variation in assembly times
7) Individual subsidiaries do not have resources and knowledge to work on an advanced planning and scheduling tool
8) Too many uncertainties, variation and restrictions influence the assembly process
9) Subsidiaries have to create complex assembly plan manually
10) The current assembly planning and scheduling approach cannot deal with uncertainties varations and restrictions
11) Unrealistic assembly schedule that does not meet desired assembly times and volumes
12) Assembly workers do not work efficiently (too much idle and over time)
13) Assembly planning and scheduling approach is not proactive and dynamic
14) Assembly schedule requires too often modification
15) Percentage of on-time deliveries is too low
16) High workload for planners at all subsidiaries
17) Too much money spent on penalties
Legend
Uncontrollable / out of scope causes
Root cause to be solved
Effects
Main effect
Cause to effect direction

Figure 1-4. Problem Cluster

There are 3 main uncertainties, variations and restrictions that make it difficult to manually create a realistic schedule. The first uncertainty is related to the unavailability of materials (4). The Supply Chain Manager of Royal Terberg Group (2022) mentions that materials are too often not available when the assembly needs to start. This is because suppliers deliver their materials too late (1). The absence of materials results in extra work during the assembly or after the assembly. Selecting a different supplier cannot resolve the problem since the scarcity of materials is a problem for all potential suppliers of Terberg. Besides, there are no alternative suppliers for a substantial number of products. Currently, the Data Science Department is working on a traffic light system regarding the availability of materials. The planner sees the presence of the (essential) materials and the delivery date when it is absent. This system is currently not in use.

Second, Terberg only has a limited amount of available working hours (5). The assembly workers of Terberg only work and are willing to work during the daytime. Besides, there is a limited number of assembly workers available. Therefore, the restriction is the availability of assembly staff (2). At each subsidiary, there are only a few employees that have the right skill set to work at different assembly. This means that in general, an employee can only work on in a specific assembly stage. The overall skilled employees assist the other workstations when problems occur. There is also a difference in skill level within an assembly stage. Certain more experienced employees perform more complex tasks than others. Therefore, there is a limited number of products per product type that each assembly line can assembly. A production employee at Terberg Benschop (2022) mentions that currently the schedule does not consider the available number of (rightly skilled) assembly workers.

Third, there is a large variation in assembly times between vehicles or vehicle systems (5). Terberg offers a wide range of vehicles and vehicle systems to its customers. Furthermore, Terberg offers more than 1000 additional options per assembly line (3). This results in different assembly times for each individual product in each assembly stage. The historical assembly times are known. Therefore, Terberg can make a proper estimation of the assembly time of each vehicle or vehicle system. However, the planner has difficulties with aligning and balancing these varying assembly times at every stage of the assembly line.

As earlier mentioned, the current assembly planning and scheduling approach cannot deal with these uncertainties, variations, and restrictions since it is not dynamic (13). Both the Production Manager of Terberg Benschop and Machines (2022) mention that even a basic assembly scheduling approach that considers the variation in products, uncertainties, and restrictions results in a significant improvement in assembly performances. Currently, there is an absence of the integrated usage of available assembly data e.g., availability of materials, availability of (skilled) workers and assembly times.

The planner at each subsidiary creates the complex assembly schedule manually (9). According to the planner of Dennis Eagle (2022), 80% of the time they are busy with manually checking if an assembly schedule is feasible. They do this with numerous calculations in different Excel sheets. The assembly subsidiaries of Terberg do not have an advanced scheduling tool because they do not have the resources and knowledge to work on it (7). Terberg aims for a generic assembly scheduling approach for the subsidiaries. Therefore, they do not have to create a different approach for each subsidiary and make use of economies of scale.

We choose the core problem for the research based on the problem cluster and the principles of the MPSM of Hans Heerkens & Van Winden (2012). This results in the following core problem:

"The assembly scheduling approach at Terberg's subsidiaries does not lead to realistic scheduling since it is not dynamic, predictive, and proactive. Royal Terberg Group wants to develop a generic tool that supports the assembly scheduling at each subsidiary."

The discrepancy between the norm and the reality of the problem is the absence of a scheduling approach that considers the variation, uncertainties, and restrictions at each assembly subsidiary of Terberg.

1.2.2 Research Scope & Objective

The scope of the research is limited to 3 assembly subsidiaries of Terberg. The research focuses on the Terberg Benschop B.V., Terberg Machines B.V. and Dennis Eagle LTD subsidiaries. The 3 facilities produce various products. However, there are similarities between the assembly locations in terms of process steps and flow.

Terberg has assembly subsidiaries that depend on each other. This means that one subsidiary produces parts for another subsidiary. These dependencies are not within the scope of this research. The research focuses on each individual assembly subsidiary.

There are multiple variations, uncertainties, and restrictions in the assembly process. The research focuses only on the variations and uncertainties with the strongest impact on the assembly process according to the assembly employees of Terberg. The variation is in the due date, the assembly times and needed parts of individual products. The uncertainty is about the availability of materials. Additionally, each assembly line has its own restrictions regarding the availability of workstations and allowed sequences.

Based on the scope and the problem identification we formulate the following research objective:

“Develop a dynamic scheduling approach that considers the variations, uncertainties, and restrictions related to the assembly process and that is suitable for the assembly subsidiaries of Royal Terberg Group to decrease the makespan and increase the number of on-time deliveries.”

1.3 Research Approach

From the core problem, research objective, and scope, we derive the following main research question:

"What scheduling approach can be adopted by the assembly subsidiaries of Royal Terberg Group to effectively enhance the quality of the schedule, while considering the variations, uncertainties, and restrictions?”

Together with the employees of Terberg, we determine the key performance indicators (KPIs) that best represent the quality of the schedule. By involving the employees in the research process, we ensure that their expertise and perspectives are incorporated into the evaluation of schedule quality and the determination of KPIs. This approach enhances the relevance and significance of the KPIs and ensures that they align with the specific needs and goals of the assembly subsidiaries of Royal Terberg Group.

We divide the research approach into 6 phases. Figure 1-5 shows the research phases. Each phase has a single research question and multiple sub-questions. We describe for each phase what information we need and how we gather this information.

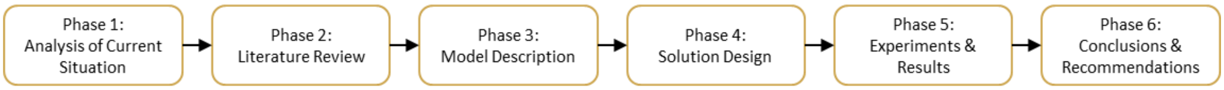


Figure 1-5. Research Phases

Phase 1: Analysis of Current Situation

We answer the following research question in Phase 1:

“What is the current assembly, planning and scheduling situation at Terberg?”

To answer this research question, we answer the following sub-questions:

1.1 What do the assembly processes at the subsidiaries of Terberg look like?
1.2 What does the planning and scheduling process currently look like?
1.3 What are the uncertainties, variations, and restrictions for the assembly planning and scheduling process at each subsidiary?
1.4 What are the current and (potential) future performance indicators?

In Phase 1, we analyse the current assembly processes of each subsidiary at Terberg. Additionally, we examine the current planning and scheduling approaches at Terberg. We discuss Phase 1 in Chapter 2. A clear overview of the current assembly, planning and scheduling processes is essential to improve the scheduling approach. We visit the assembly subsidiaries within the scope to see what and how they assemble the products. We observe the assembly process by taking a tour through assembly halls. In addition, we interview assembly workers and managers to make ourselves also familiar with the wishes and problems that are present at the assembly subsidiaries of Terberg. In this phase, we also determine the objectives and restrictions for the assembly schedule. This information is the basis for the creation of the scheduling model. Besides, we identify the performance measures of Terberg and analyse the current values.

Phase 2: Literature Review

We answer the following research question in Phase 2:

"What relevant information can we use from the literature to understand the problem at Terberg and formulate an effective solution approach to solve it?"

To answer this research question, we answer the following sub-questions:

2.1 How are the assembly lines and scheduling problems of Terberg known in the literature?
2.2 What methods are available in the literature to solve the problem?

Chapter 3 discusses the second phase of the research. In this phase, we conduct a literature review. This includes literature related to the present planning problems at Terberg. We classify the problem to get a better understanding in which direction the potential solution can be found. Besides, we analyse frameworks and other solution approaches from the literature to solve our specific planning and scheduling problems.

Phase 3: Model Description

We answer the following research question in Phase 3:

"How can we systematically describe and model the scheduling problem of Terberg?".

To answer this research question, we answer the following sub-questions:

3.1 What are the different and similar aspects between the assembly subsidiaries?
3.2 How can we model the scheduling problem of Terberg including the variations, uncertainties, and restrictions?
3.3 Which objectives are most suitable to evaluate a schedule?

Chapter 4 describes the specific scheduling problem. With the use of the literature from the previous chapter we characterize the assembly lines. In addition, we formulate assumptions and simplifications. These assumptions and simplifications help simplifying the solution analysis without decreasing the quality of the result. Thereafter, we formulate the model to systematically define the planning and scheduling problem of Terberg. The model includes the variation, uncertainties, and restrictions. Lastly, we determine in this phase the objectives to evaluate a schedule.

Phase 4: Solution Design

We answer the following research question in Phase 4:

"Which alternative solutions approaches are suitable to solve the scheduling problem of Terberg?"

To answer this research question, we answer the following sub-questions:

4.1 Which approach from the literature can we use to solve the scheduling problem of Terberg?
4.2 What are the restrictions, objectives, and requirements of Terberg for the model?
4.3 How can we adapt the methods to the scheduling problem situation of Terberg?

In Phase 4 we formulate approaches to solve the specific scheduling problem of Terberg. We discuss this phase in Chapter 5. Together with the most relevant models from the literature review and the data of Terberg we develop suitable scheduling approaches. We also use an expert panel to validate the model(s). This ensures that the model fulfils the requirements of the subsidiaries.

Phase 5: Experiments & Results

We answer the following research question in Phase 5:

"Which alternative solutions approach performs best compared to each other and how does the best solutions approach perform under different experimental settings at Terberg?"

6.1 What kind of trade-offs in KPIs are present?
6.2 What are the differences between developed approach and the current approach?

In Phase 5 we implement the solution design and create a proof-of-concept of the planning tool. Chapter 6 describes Phase 5. We use the real-life available assembly data to test the designed solution models. Furthermore, we introduce alternative models to observe how the system performs under varying experimental settings. We use an expert panel to validate the generated solutions. This ensures that the model fulfils the requirements of the employees.

Phase 6: Conclusions & Recommendations

In the last phase, we conclude the research. Chapter 7 discusses the answer to the overall research question. Besides, we present our recommendations and conclusions for Terberg. In the end, we present the limitations in the research and suggest further research topics.

2 Current Situation

The goal of this chapter is to answer the first research question:

"What is the current assembly, planning and scheduling situation at Terberg?"

Section 2.1 describes the assembly lines of the subsidiaries within the scope of the research. Section 2.2 discusses the current planning and scheduling process. Additionally, it gives insight into the various part types and the impact that shortages have on the (post-)assembly processes. Section 2.3 describes the objectives and restrictions to the assembly, planning and scheduling process. Section 2.4 concludes this chapter.

2.1 Assembly Lines

This section describes the different assembly subsidiaries of Terberg that are within the scope of the research. Section 2.1.1 presents the assembly halls and process of Terberg Benschop B.V. Section 2.1.2 shows the assembly hall and process of Terberg Machines B.V. Lastly, Section 2.1.3 describes the assembly hall and process of Dennis Eagle LTD. Appendix A shows the product-mix and assembly statistics per assembly line.

2.1.1 Terberg Benschop B.V.

This section presents the assembly line of Terberg Benschop B.V. The Terberg Benschop B.V. subsidiary is in the village of Benschop. This facility assembles special vehicles to transport containers. Customers deploy these vehicles in ports, airports, distribution centres and other industrial sites all over the world. Terberg Benschop B.V. works with two independent assembly halls, assembly line 1 (AS1) and assembly line 2 (AS2). Figure 2-1 shows an overview of the facility and the location of both halls. The warehouse of this subsidiary is in a different hall.

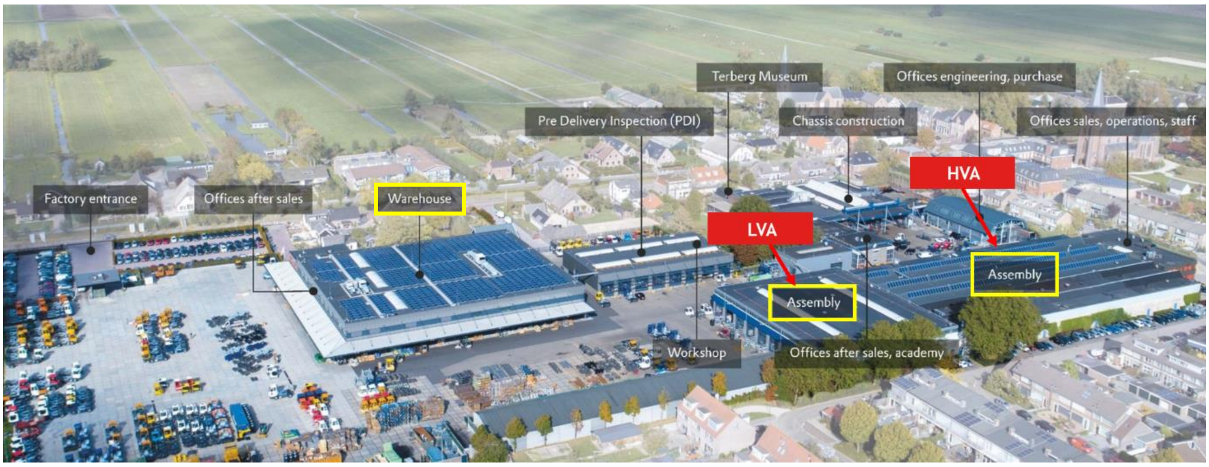

Terberg Museum
Offices engineering, purchase
Pre Delivery Inspection (PDI)
Chassis construction
Offices sales, operations, staff
HVA
Factory entrance
Offices after sales
Warehouse
LVA
Assembly
Assembly
Workshop
Offices after sales, academy

Figure 2-1. Overview Terberg Benschop B.V.

Assembly Layout AS1

This assembly line assembles the vehicles with less complexity. AS 1 consists of 6 main stages and a total of 17 substages for preparation. Each main stage has room for 2 vehicles. Figure 2-2 presents the layout of AS1. Both vehicles move to the next stage simultaneously after a cycle time. The cycle time depends on the number of vehicles Terberg Benschop B.V. plans to assemble in a week. This means that the cycle time can change every week. The available working time is normally 38 hours per week. Equation 2-1 shows the calculation of the cycle time. There is still a difference in complexity between different vehicle types. As a result, there are restrictions that prohibit specific product types from coexisting within the same stage simultaneously. Appendix A shows the product-mix and assembly statistics of AS1.

Equation 2-1. Cycle Time Calculation

$$T = \frac{T_a * M}{D}$$

T = cycle time required to meet demand

T_a = net time available to work

M = number of parallel workstations per stage

D = vehicles per week

Each (sub)stage works with a different number of employees. There are also flexible workers walking around. They help the employees at (sub)stages where the expected remaining assembly time surpasses the remaining cycle time to maintain the flow.

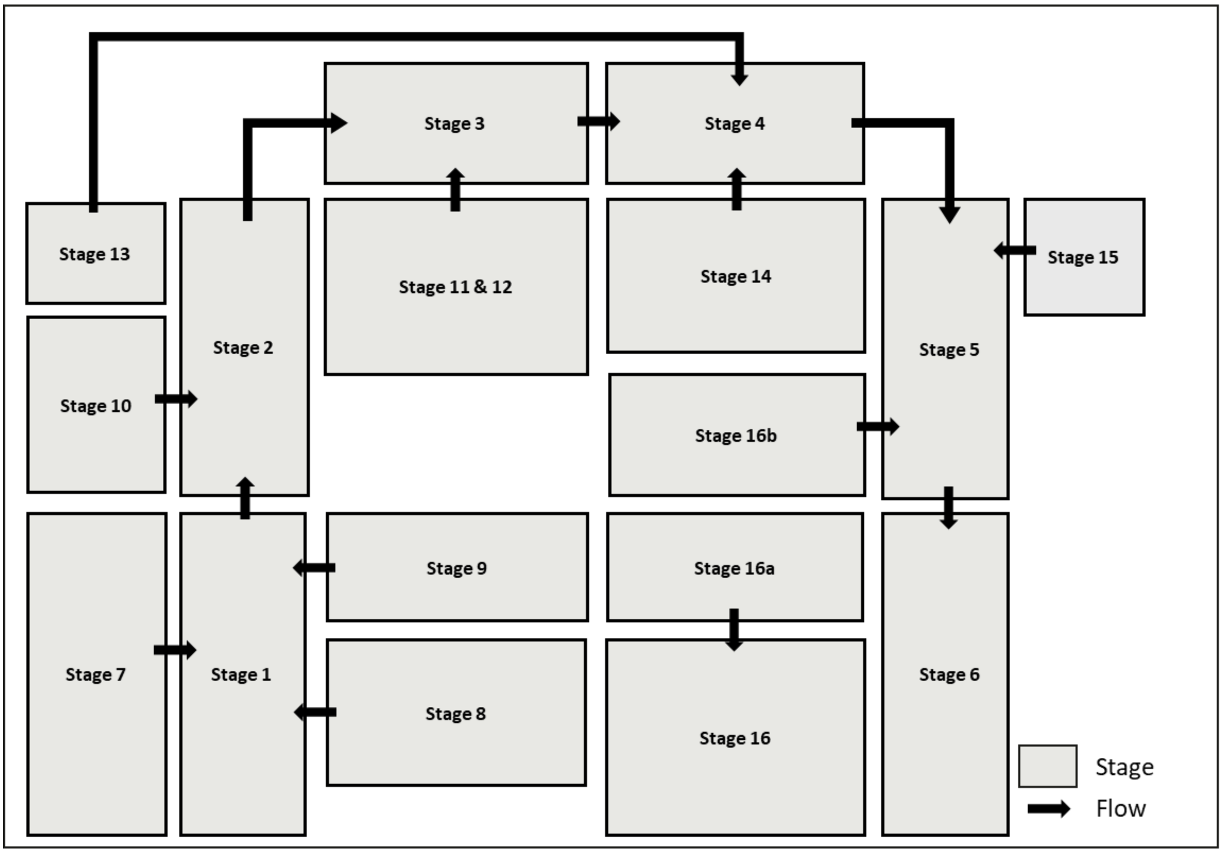


Figure 2-2. Assembly Hall Layout AS1

Figure 2-3 presents the workflow including the main- and substages of AS1. A stage can only start when the previous main stage and the needed substages are finished.

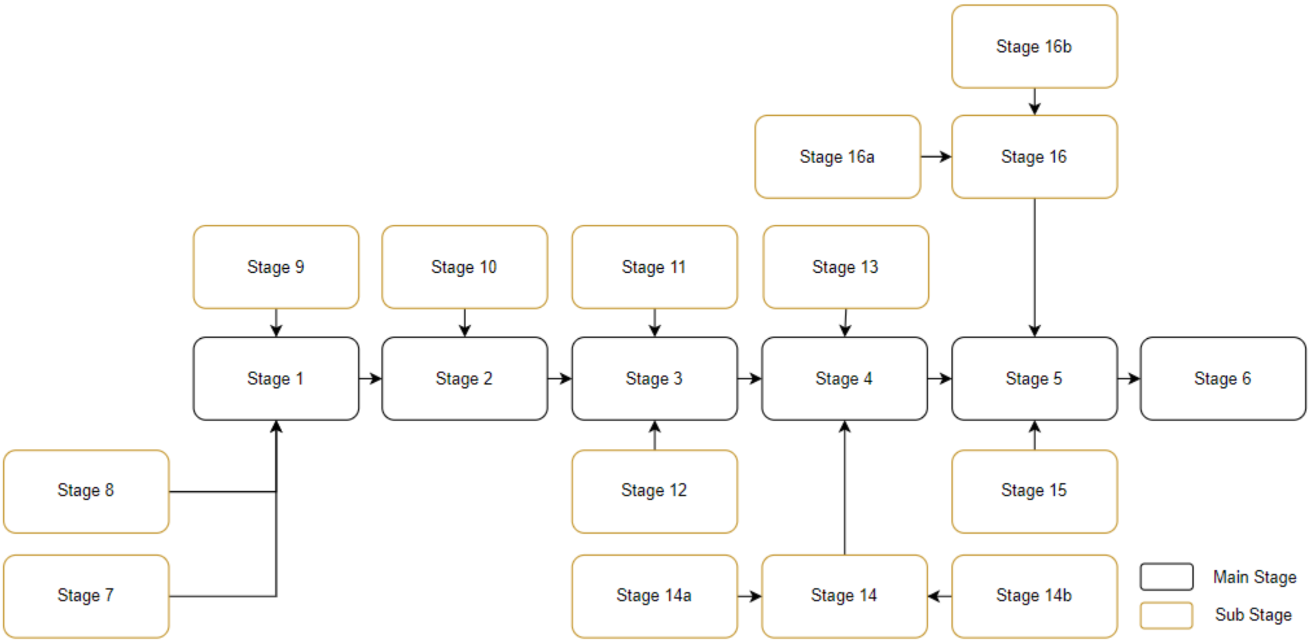


Figure 2-3. Workflow AS1

Assembly Layout AS2

In the second hall, Terberg Benschop B.V. assembles complex vehicles. There is even more variety between the vehicles and assembly times in this hall. Figure 2-4 shows the layout of the second hall. AS2 consists of 3 main stages and 10 substages for preparation. Each main stage has room for a fixed number of vehicles. The 3 main stages have 3, 3 and 4 workstations, respectively. Certain vehicle types can only be processed on a specific workstation. These workstations are large enough or have the right equipment to assemble certain vehicle types. The vehicles move independently from each other in this assembly hall. Appendix A shows the product-mix and assembly statistics of AS2.

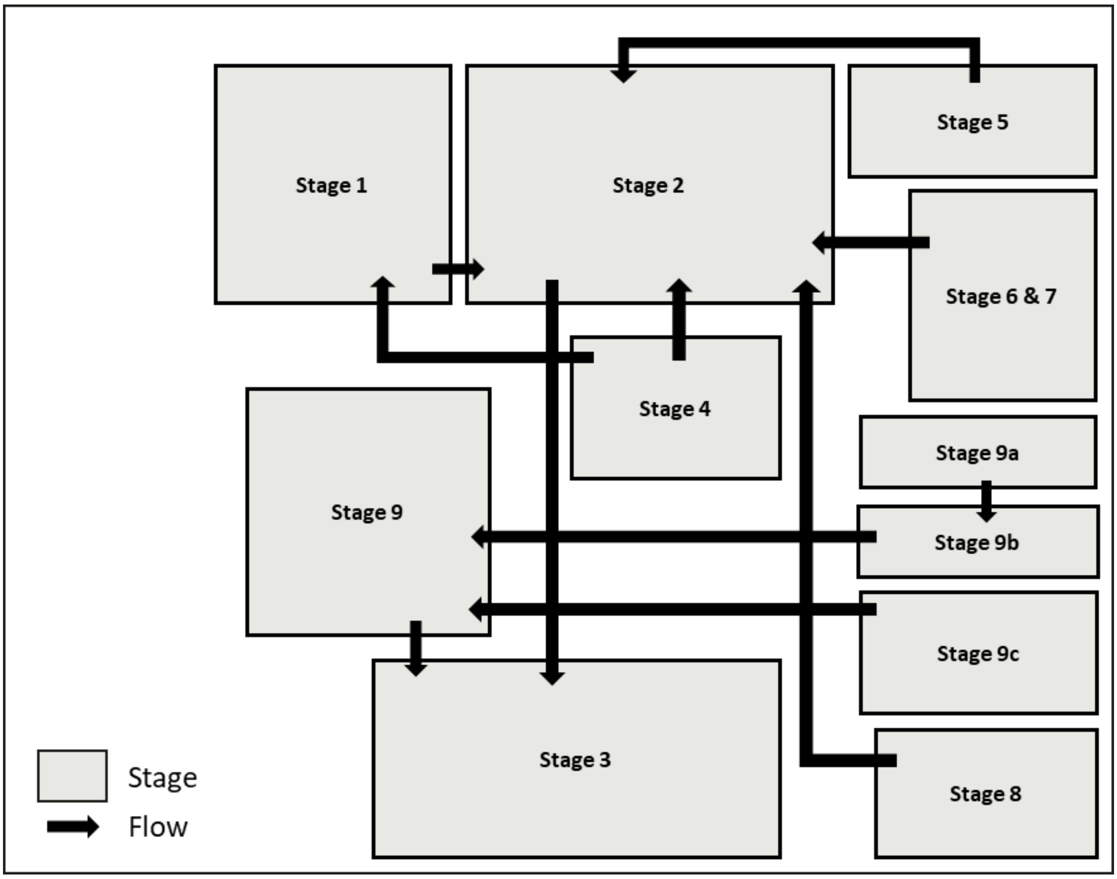


Figure 2-4. Assembly Hall Layout AS2

Each (sub)stage works with a different number of employees, who can only work in a single stage. However, workers can work on different workstations within the stages. This means that multiple workers can work on a single vehicle to decrease the throughput time. Figure 2-5 shows the workflow including the main- and substages of AS2. A stage can only start when the previous main stage and the needed substage are finished.

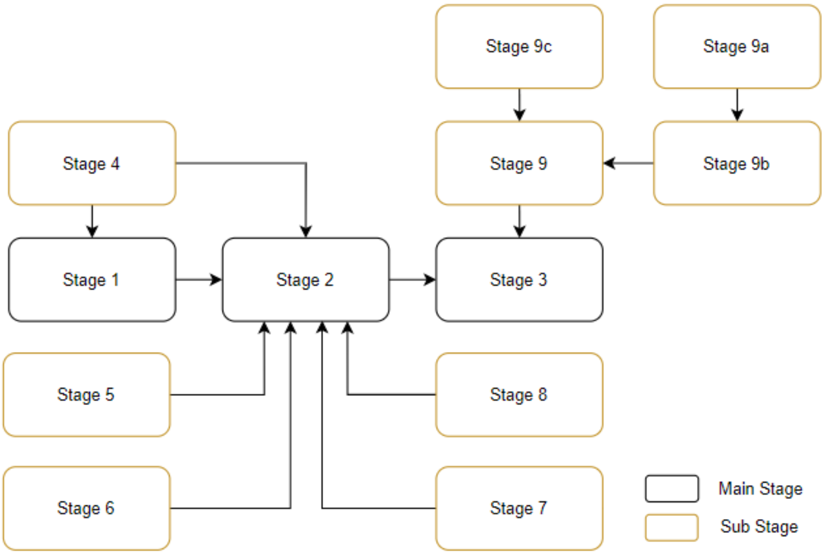


Figure 2-5. Workflow AS2

2.1.2 Terberg Machines B.V.

The Terberg Machines B.V. subsidiary is in IJsselstein. This subsidiary assembles bin lift systems. These are sold through the Royal Terberg Group. This means Terberg Machines B.V. sells the bin lift systems to other subsidiaries.

Assembly Layout AS3

AS3 consists of 6 main assembly stages and 4 preparing substages. Each main stage has room for a different number of bin lift systems. The bin lift systems move independently from each other through the facility. The subassembly must be ready to start the linked main assembly. Figure 2-6 shows the layout of the assembly hall. Certain bin lift system types need specific workstations in an assembly stage. Appendix A shows the product-mix and assembly statistics of AS3.

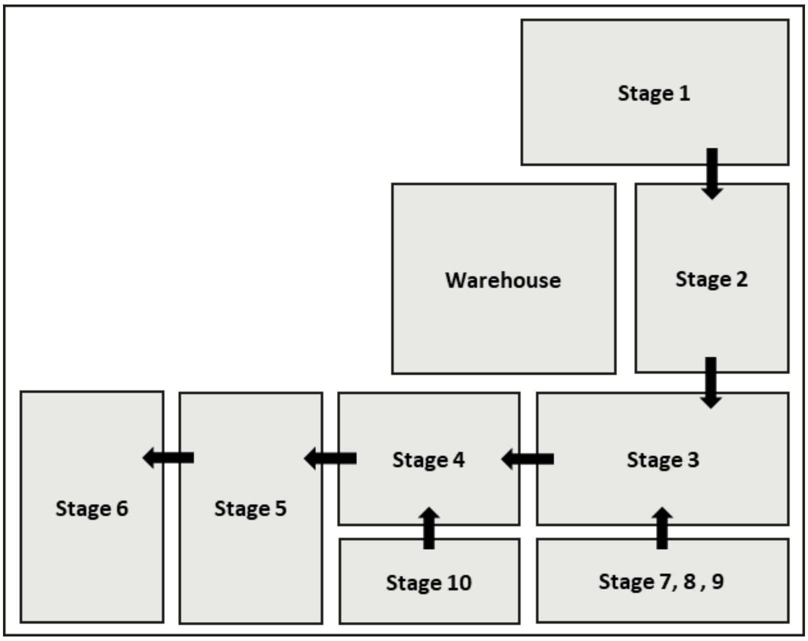


Figure 2-6. Assembly Hall Layout AS3

Figure 2-7 visualises the workflow including the main- and substages of the product route. An assembly in a stages can only start when the previous assembly in the main stages and the required substages are finished.

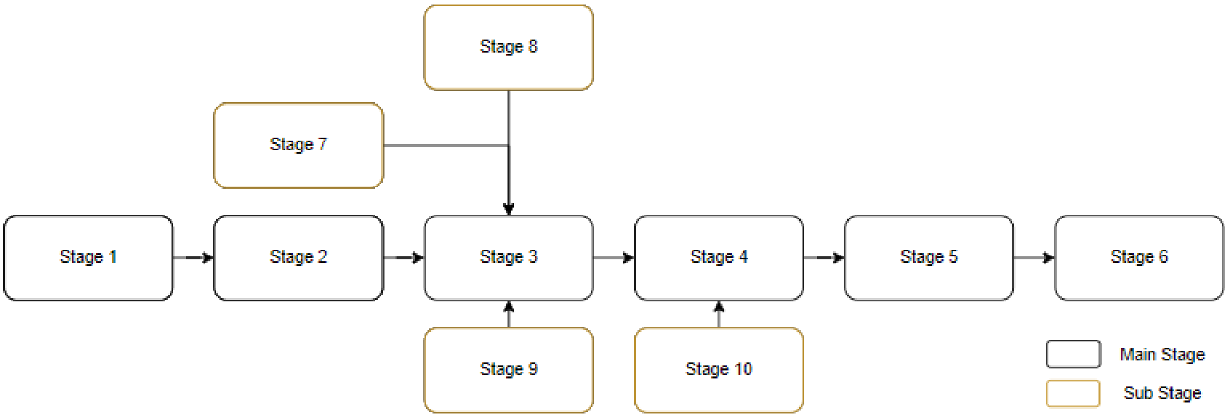


Figure 2-7. Workflow AS3

2.1.3 Dennis Eagle LTD

The Dennis Eagle LTD subsidiary is in Warwick, England. This subsidiary of Terberg assembles Refuse Collection Vehicles (RCV). They produce 28 vehicles per week. Figure 2-8 shows the layout and the assembly lines of the manufacturing facility. There are three manufacturing halls called Unit 1, Unit 2, and Unit 3. Each unit has one or multiple assembly lines with single workstations in series.

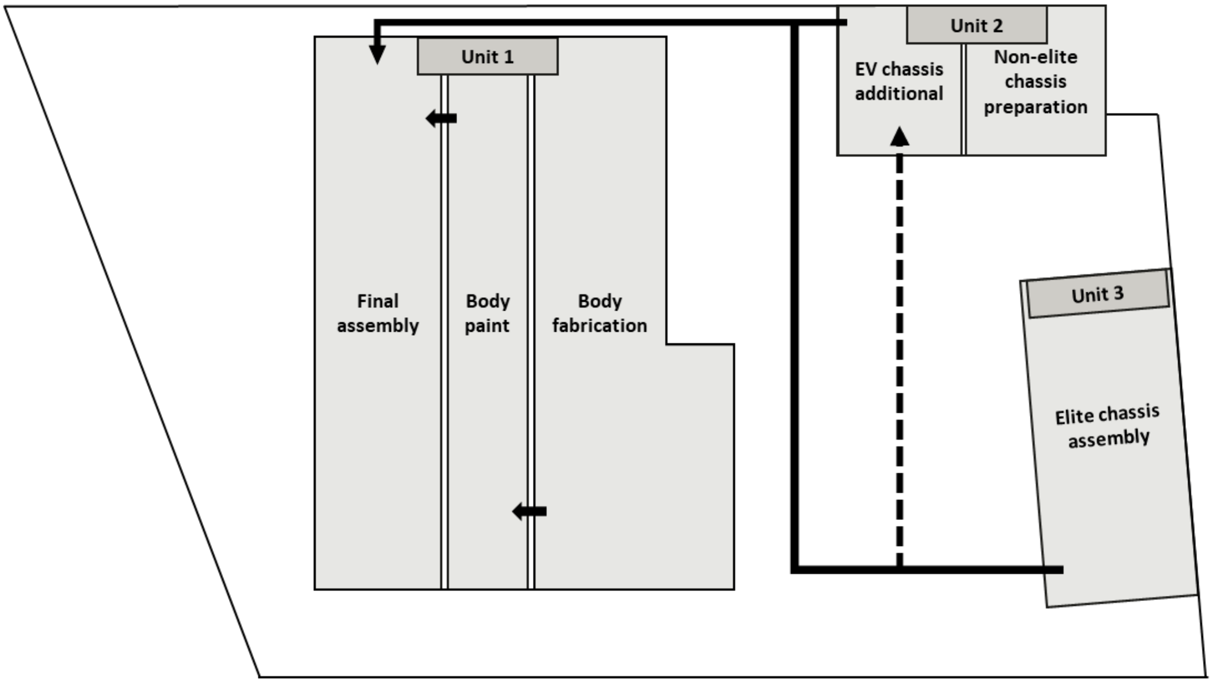


Figure 2-8. Assembly Hall Layout AS4

The RCV consist of 4 main subassemblies: the chassis, cab, body, and hopper. Figure 2-9 shows the structure of the RCV. The chassis is the frame on which to build the rest of the vehicle. The cab is the area where the driver sits. The body of the RCV stores the collected waste. The hopper transfers the waste from a waste container to the body. It also pushes the waste to the back of the body.

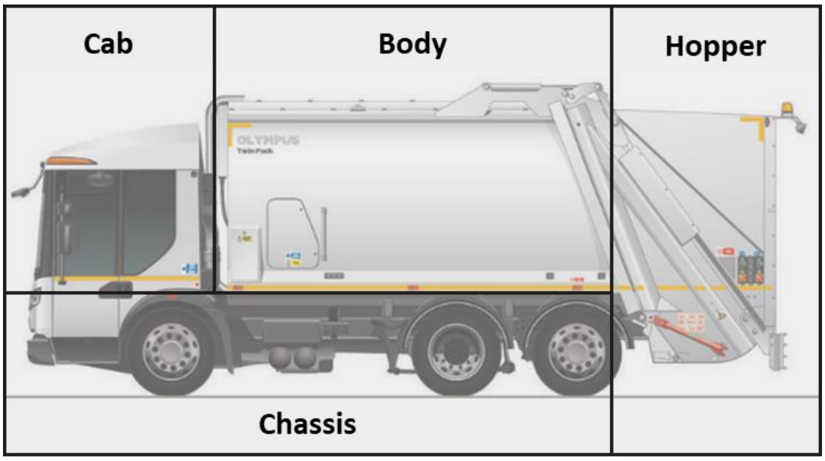


Figure 2-9. Structure Refuse Collection Vehicle

2.2 Planning and Scheduling Process Description

This section describes the planning and scheduling process. The 3 assembly facilities create their own assembly plan. However, the planning and scheduling processes of the subsidiaries are almost similar. We divide the total planning and scheduling process into two processes, the long-term planning and short-term scheduling process. Section 2.2.1 discusses long-term planning and Section 2.2.2 describes the short-term scheduling process. Section 2.2.3 gives insight into the different part types and the impact that shortages have on the (post-)assembly processes.

2.2.1 Long-Term Planning

The long-term planning process starts with a quotation request from the customer. The sales department of each subsidiary of Terberg negotiate with the customer regarding pricing, preferred delivery week and potential delivery penalties. In most cases, the customer wants their product at the earliest opportunity. The lead time for the customers differs per assembly subsdiary. The complexity of the product influences the lead time. Figure 2-10 presents the long-term planning and ordering process at Terberg.

Currently, the delivery week depends on the assembly capacity per week. Each vehicle (system) is assigned a certain workload value. Each subsidiary deals with a maximum total workload per week. The workload value is directly influenced by the complexity of the vehicle or system. Furthermore, each subisidary has a predetermined limit on the maximum number of products they can assemble per product type. Exceeding this limit leads to an overload of work at particular stages.

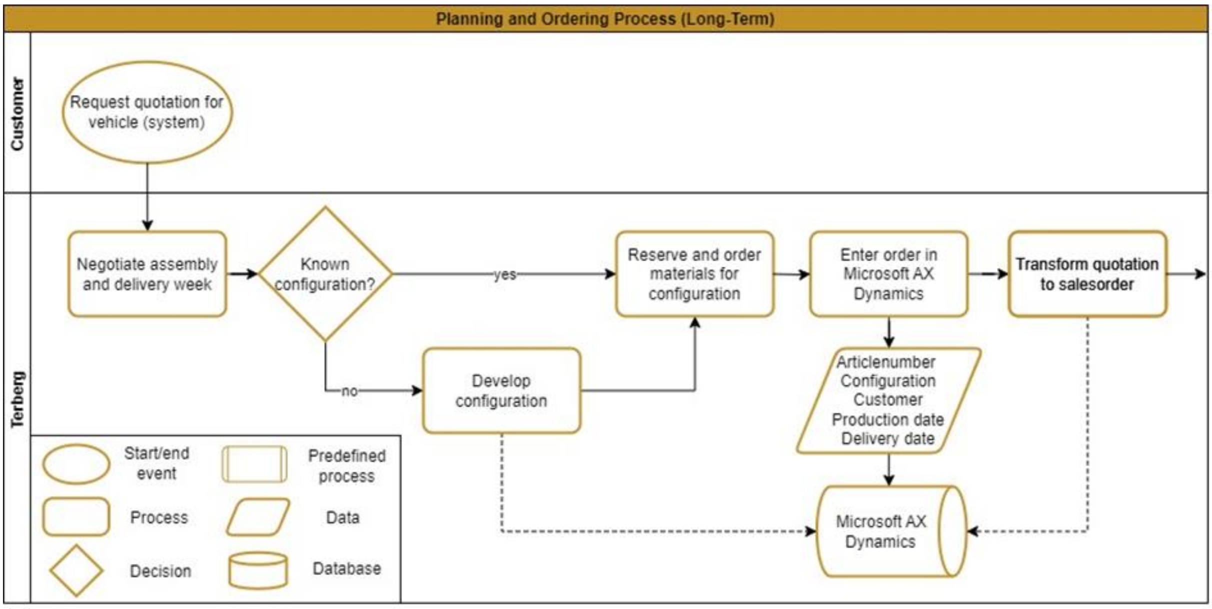

Planning and Ordering Process (Long-Term)
Customer
Terberg
Request quotation for vehicle (system)
Negotiate assembly and delivery week
Known configuration?
yes
no
Reserve and order materials for configuration
Enter order in Microsoft AX Dynamics
Transform quotation to salesorder
Develop configuration
Articlenumber
Configuration
Customer
Production date
Delivery date
Microsoft AX Dynamics
Start/end event
Predefined process
Process
Data
Decision
Database

Figure 2-10. Flowchart of Long-Term Planning and Order Process

Table 2-1 shows an example of the capacity restriction of Terberg. In this example, the assembly line can handle a workload value of 50 per week. In addition, there is a maximum number of products per product type that can be assembled. For example, Terberg aims to assemble a maximum of 5 products of product type 7 per week. When the planner exceeds these capacity restrictions, a warning occurs. Sometimes it is necessary exceed the capacity restriction to meet due dates and avoid penalties.

Table 2-1. Example of Long-Term Weekly Capacity Restriction

Week Nr.	**37**	**38**	**39**	**40**	**41**	**42**
Product Type 1	0	0	0	1	1	2
Product Type 2	4	4	4	3	4	4
Product Type 3	0	0	0	0	0	0
Product Type 4	5	5	6	8	4	5
Product Type 5	25	22	21	18	27	23
Product Type 6	2	2	2	2	0	0
Product Type 7	7	7	6	7	5	5
Product Type 8	7	7	8	8	7	9
Product Type 9	0	0	0	0	1	1
Product Type 10	0	1	0	1	1	1
Total Products	**50**	**48**	**47**	**48**	**50**	**50**
Workload	50	48	47	48	48	48
Capacity	50	50	50	50	50	50
Available Workload	0	2	3	2	2	2

Terberg engineers the configuration when the configuration is new. Thereafter, the employees of Terberg enter the order in Microsoft Dynamics and reserve the parts. The reservations trigger the need for parts. This results either in externally purchasing or internally producing the parts. The order and production quantities also depend on economical order quantities. This means that it is not financially beneficial for Terberg to purchase an item individually every time it is required. Terberg knows the delivery date of parts.

Microsoft Dynamics

Terberg uses the Enterprise Resource Planning (ERP) software package Microsoft Dynamics. The employees of Terberg enter information in and retrieve information from Dynamics during the entire planning and order process. In Dynamics, employees can reserve material for a specific order. The software triggers a need for each part connected to an order configuration. This result in purchasing requests at the purchasing department.

In Microsoft Dynamics, the employees see where the part is stored and where it is used. Besides, they see the current inventory and the expected inventory over the coming year. This inventory expectation uses the expected future deliveries and assembly.

2.2.2 Short-Term Planning and Scheduling Process

The short-term planning and scheduling process starts 4 weeks before assembly. Figure 2-11 shows the short-term planning and scheduling process of Terberg. In the first step, a shortage check takes place. The planner reschedules an order when an urgent shortage occurs. A shortage is urgent when the assembly cannot continue. In addition, Terberg also checks if there is an option to speed up the purchase or produce the part internally on time. The sales order becomes a production order when there are no urgent shortages present.

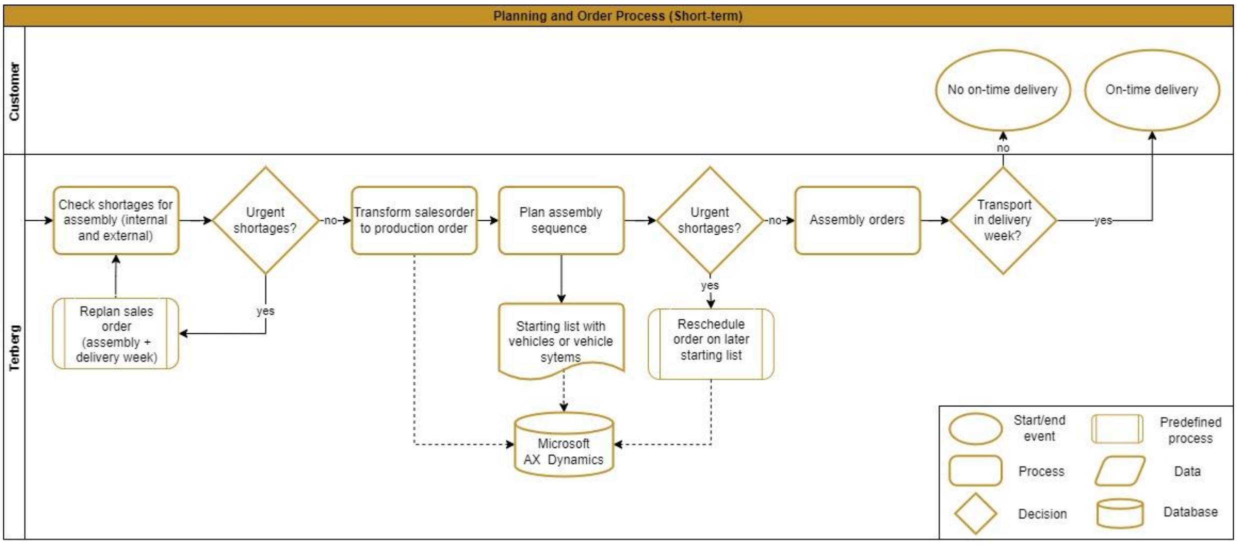

Planning and Order Process (Short-term)
Customer
Terberg
No on-time delivery
On-time delivery
no
Check shortages for assembly (internal and external)
Urgent shortages?
no
Transform salesorder to production order
Plan assembly sequence
Urgent shortages?
no
Assembly orders
Transport in delivery week?
yes
yes
Replan sales order (assembly + delivery week)
yes
Starting list with vehicles or vehicle sytems
Reschedule order on later starting list
Microsoft AX Dynamics
Start/end event
Predefined process
Process
Data
Decision
Database

Figure 2-11. Flowchart of Short-Term Planning and Order Process

Next, the planner schedules the production order on the day level for a week. This results in a starting list with the sequence of products. Currently, the planner sequences the vehicles or vehicle systems by spreading vehicles with different degrees of complexity over the week. Terberg expresses the difference in complexity in the total assembly time. The planner mixes the vehicles or vehicle systems with longer and shorter assembly times. Table 2-2 shows an example of a starting list for Monday, Tuesday, and Wednesday. Terberg aims to create a starting list that also consider the starting list of the preceding week. The products in the end of the preceding starting list are still in the assembly line when the current week starts.

Table 2-2. Starting List with Assembly Orders

Monday		**Tuesday**		**Wednesday**	
Order	**Assembly Time**	**Order**	**Assembly Time**	**Order**	**Assembly Time**
Order 2	5.5	Order 13	5.25	Order 21	5.5
Order 6	3.5	Order 17	5.5	Order 26	3.5
Order 19	3.5	Order 1	3.5	Order 14	2
Order 32	3	Order 3	5.5	Order 30	5.5
Order 4	3.75	Order 33	2	Order 40	3.75
Order 12	3.5	Order 37	3.5	Order 36	3.5
Order 25	7	Order 12	3.5	Order 28	7
Order 15	2	Order 19	3.75	Order 39	3.5
Order 7	2	Order 8	2	Order 9	2

The planner uses experience and gut feeling to create the starting list. They try to manually determine the start and finish times to align the assembly process with the use of basic calculations in Excel. A final shortage check takes place one week before assembly. The planner reschedules the order on a later starting list when there is still a critical part missing. Terberg delivers an order as agreed when they produce the order in the predetermined correct week and transport it in the correct week.

2.2.3 Parts & Shortages

This section provides an overview of the types of parts Terberg has and offers insight in the effect of shortages. Terberg requires parts for each vehicle or vehicle system, consisting of both product-specific and multi-purpose components. The product-specific components are customized to meet specific dimensions, properties, and/or colour requirements, such as a longer chassis, a more powerful engine, or a cabin in a particular colour. Additionally, Terberg utilizes common components across its products, sourced from a shared inventory and allocated on a first-come-first-serve basis to the product that requires it. The Bill of Materials (BOM) contains all parts that the employees need to assemble all products in a period. There are supply moments to replenish the inventory. Terberg knows when these supply moments are and the supplied quantity. A potential shortage arises when the initial inventory levels fall short of meeting the required quantities specified in BOM and there are no replenishment quantities available to make up for the shortfall.

Section 1.2.1 states that each subsidiary deals with an increasing absence of parts. Each shortage can have varying impacts on the assembly line. Terberg uses a scale from 1 to 4 to categorise each part that is needed for the assembly. Table 2-3 shows the scores of the impact of a shortage. The higher the score the more impact the shortage has on the (post)-assembly process.

Table 2-3. Impact of Shortages Score

Impact Score	Description	Examples
1	The missing part is sent later to the customer	Clutch head, stickers, towbar, or tires
2	The missing part is exchangeable with other parts or easy to attach afterwards	Cables, chair, computer, pipes, switches, or steering wheel
3	The missing part is hard to attach afterwards	Battery, display, filters, fluid tanks or steering cylinder
4	The missing part blocks the assembly completely	Cabin, chassis, or engine

The first category includes the parts with the least impact. The employees easily attach the part later or sent it to the customer when the product already left the assembly facility. The second category contains parts that are exchangeable with other parts with the same functionality or are easy to attach afterwards. The third category contains parts that are hard to attach and need significant amount of work to assemble afterwards. The fourth category contains parts that are essential for the assembly. The assembly cannot continue when one of these parts are not present.

2.3 Objectives and Restrictions

The employees within the different departments at Terberg mention a wide range of different objectives. Section 2.3.1 presents these objectives. The objectives of the departments can conflict with each other. This leads to restrictions to the schedule. Section 2.3.2 discusses these restrictions. Section 2.3.3 presents the current performance measures. This section also discusses potential performance measures for the new scheduling approach mentioned by the employees.

2.3.1 Objectives

Through interviews with Terberg employees, it becomes clear that various employee groups have distinct scheduling and assembly-related objectives. To better understand and organize these objectives, we categorize them based on department or employee function.

Production Managers

The most important objective for the Production Managers is to sell as many vehicles or vehicle systems as possible. The assembly of these vehicles or vehicle systems must be done with the least number of resources used. Therefore, it is important to them that there is as little idle time and overtime as possible. Besides, they aim minimize the number of total penalties. This results in a loss of turnover. Another crucial objective is to minimize the amount of time a vehicle (or system) spends on the assembly line and within their facility. The vehicles or vehicle systems occupy the limited space that is available at the facilities and increase the amount of working capital.

Planner

The goal of the planner is to create a realistic assembly schedule. This is an assembly schedule that is feasible with the available resources. The planner also aims for as less changes to the assembly plan as possible. Most of the time, the unavailability of resources is the cause of rescheduling the vehicles or vehicle systems. Especially, in recent years when the availability of materials and workers becomes more and more uncertain. Therefore, another objective is to have minimal (re)scheduling time. Currently, the rescheduling time is high since the planner must do all the manual calculations again.

Production & Warehouse

The Production Employees want feasible assembly volumes. This means that they assemble all the vehicles with the available workforce and time. Their objective is to minimize overtime work. Another objective is to have a constant flow of vehicles. The workload needs to be equally spread over the week. Their goal is to reach their due dates and cycle times as often as possible. They want a mixture between complex vehicles with easier vehicles each week.

The Warehouse Employees pick the needed materials for each assembly in the assembly line. An objective of the employees in the warehouse is equal workloads during the week. A predictable and realistic assembly schedule shows warehouse employees when demand peaks are expected.

Sales

The primary goal of the Sales Department is to maximize sales while simultaneously providing customers with the shortest possible delivery times. Furthermore, they strive for a high percentage of on-time deliveries to avoid incurring penalties. Additionally, the Sales Employees aim to provide customers with a diverse range of options for vehicles and vehicle systems. They offer the customer a wide range of products and additional customization options. This enhances the customer satisfaction. However, an increase in product variations results in varying assembly times. This makes scheduling more difficult.

Purchasing

The Purchasing Department primary objective is to ensure that timely availability of necessary materials and parts. To achieve this, they create risk profiles for suppliers, which include lead times and the variance between requested and actual delivery dates. The Purchasing Department prioritize suppliers with favourable risk profiles. Moreover, they seek to establish multiple suppliers for similar whenever feasible. However, dual sourcing is not always possible for complex components.

2.3.2 Restrictions

Due to conflicting objectives among different departments and employee functions, it becomes challenging to meet the goals of all parties involved. Therefore, these conflicting objectives place constraints on the assembly line and planning process. The planner must consider these constraints to formulate a feasible schedule.

Workload Restriction

Currently, the most critical concern is managing the workload effectively due to a shortage of assembly employees. There are not significantly more assembly employees available. Terberg solely assembles during daytime hours. Moreover, the option for work outside regular working hours is restricted, with only minimal overtime work permitted. The planner constructs the weekly assembly schedule considering the workload limitations. Consequently, each assembly line can only accommodate a maximum workload to ensure the assembly of vehicles and vehicle systems proceeds smoothly.

Skill & Workspace Restriction

Each assembly employee has a different set of skills. The skillset determines which tasks an employee can perform. Therefore, the total complexity of vehicles or vehicle systems at a certain moment in time cannot exceed the total workload of the available fixed workers and potential flexible workers. This means that too many complex vehicles or vehicle systems at once are not allowed.

There are also only a limited number of workstations and workspaces available to produce vehicles and vehicle systems. The workers can only work on a fixed number of workstations in every stage. Additionally, certain product types can only be assembled at specific workstations.

Material Restriction

Terberg produces or purchases its materials and parts. An assembly starts when all critical parts and subassemblies are available. Problems occur at the assembly line when the planner does not consider the absence of critical parts. This delays the start time of product and potentially influence the start time of consecutive products. Some parts are not critical for the assembly. The assembly can start without the part. An employee attaches the part afterwards to the product. The amount of extra (post-)assembly time differs per missing part.

R&D Restriction

Terberg offers a wide variety of products and options. Sometimes a customer requests a new configuration. The R&D department needs to develop this new configuration before Terberg starts with the assembly of this vehicle (system). The Sales Department discusses a reasonable delivery date in this case.

2.3.3 Performance Measures

Currently, Terberg measures their performance only marginally. Terberg only looks at the assembly volumes of the subsidiaries per week. Section 1.2.1 describes that the Data Scientists encounter an insufficient on-time delivery rate and a high amount of over/idle time. The Planners also indicate an increasing amount of extra work. Terberg has not conducted a comprehensive analysis of these issues, and as a result, there are currently no indicators in place to reflect the shortcomings in meeting these objectives.

Potential Future Measures

We see that the employees of Terberg have all kinds of objectives related to assembly, planning and scheduling. However, these are currently not measured. Table 2-4 lists all the potential performance measures mentioned by the employees and describes them.

Table 2-4. Potential Performance Measures

KPI	Description
Output	Number of vehicles or vehicle systems (per week)
Idle/working time rate	Percentage of time an operation is idle/working
Overtime rate	Percentage of time workers need to work overtime
Makespan	Total assembly time of a list of products
Individual Makespan	The time a product is in the assembly line
On-time finishing rate	Percentage of jobs that is finished on-time
Finishing lateness	The lateness of a product
Shortage rate	Percentage of jobs that finish the assembly line with shortages
Shortage earliness	The time that a job is assembled earlier than the moment that the shortage is resolved
Computational time	The time that a model runs to find a good solution

The most frequently mentioned objective by the employee is to maximize output. However, the composition of the assembled products needs to fulfil the demand of the customer. In line with this, Terberg aims to minimize the percentage of idle time and overtime. Therefore, Terberg aims to produce a specific number of products in the shortest amount of time. This minimizes the makespan. There is a minimal number of buffers available between stages. Terberg indicates that the individual makespan is a potential objective to reduce the waiting time before stages.

The on-time delivery is another crucial objective for Terberg. Delivering a product to late results in a penalty. Furthermore, they strive to minimize the lateness of products that are too late. On the other hand, Terberg only starts the assembling when all the critical parts are available for assembly. There are also non-critical parts that can be assembled afterwards. However, this results in extra post-assembly time. Therefore, a potential measurement is the shortage rate. This is the percentage of jobs that finish the assembly line that do not need assembly afterwards. There is also limited space available to store not-finished products at the facility. Therefore, Terberg aims to schedule products with shortages near the supply date of the missing items.

Currently, rescheduling the products results in extra work for all departments at Terberg. Therefore, it is important to create the best assembly schedule according to the previously mentioned objected. Lastly, the computational time of (re)scheduling needs to be reasonable. This is a trade-off between the quality of the schedule and the computational time.

2.4 Chapter Summary

This section concludes the chapter. The goal of this chapter is to describe all relevant information to provide a clear answer to the following research question:

"What is the current assembly, planning and scheduling situation at Terberg?".

This research focuses on 4 assembly subsidiaries within Terberg: the AS1, AS2, AS3 and AS4. Section 2.1 describes each assembly line individually. We see similarities between assembly lines. Each assembly line has main stages with substages. These substages needs to be finished before a mainstage starts. The number of workstations per stage varies per assembly line. The assembly times and volumes differ per assembly line. At AS1 and AS4, the vehicles more simultaneously to the next stage after a certain cycle time. At AS2 and AS3, the products move individually after product specific assembly times. Furthermore, each assembly has its own mixture restrictions. These constraints prevent the scheduling of products on specific workstations or positions within the assembly sequence.

Section 2.2 describes the planning and scheduling process of Terberg. The planning and scheduling processes of the subsidiaries are identical. We divide the total planning and scheduling process into 2 processes, the long-term planning and short-term planning and scheduling process. The long-term planning process starts when a customer orders of vehicle(s) or vehicle system(s). The planner plans the assembly in a certain week based on restrictions. The violation of the restriction results in extra work for certain stages or an infeasible schedule. The planner determines the exact starting order in the short-term planning and scheduling process. Currently, they do this by equally spreading vehicles with a different degree of complexity over the week. They manually calculate start and finish times in multiple Excel files. A product is rescheduled to a different week when a crucial part is missing before the start of the assembly. The planner experiences difficulties in aligning the assembly processes. Additionally, the planner encounters challenges with considering the due dates and the absence of incoming (critical) parts.

Each department and employee function has different objectives regarding the input and output in the assembly process. These objectives can conflict with each other. This results in restrictions for the schedule. These objectives can sometimes clash with each other, leading to constraints on the scheduling process. Section 2.3 elaborates on these objectives and restrictions in detail. Terberg measures their performance marginally. They only focus on the assembly volumes per week. However, the employees list potential measures to evaluate the assembly performance.

3 Literature review

The goal of this chapter is to answer the second research question:

> *"What relevant information can we use from the literature to understand the problem at Terberg and formulate an effective solution approach to solve it?"*

This chapter provides a literature review that classifies the problem and discusses solution approaches from the literature to solve the problem. Section 3.1 classifies the business strategy, assembly line and the problem of this research. Section 3.2 describes planning and scheduling to model our research problem. Section 3.3 provides solution approaches to solve the research problem.

3.1 Problem Classification

This section categorizes the business strategy of Terberg, assembly line and problem into analytical frameworks from the literature to gain a deeper understanding of the problem and enhance our overall comprehension for the potential solution approaches from the literature. Section 3.1.1 categorize the business strategy. Section 3.1.2 classifies the assembly line of the subsidiaries within the scope of the research. Section 3.1.3 classifies our specific research problem.

3.1.1 Customer Order Decoupling Point

It is crucial for manufacturing companies to strategically align with the demands of customers. The customer order decoupling point (CODP) is getting increasing attention as an important input to the design of manufacturing operations as well as supply chains. The article of Olhager (2010) defines the CODP as the point in the material where the product is tied to a specific customer order.

Companies tend to implement a CODP to increase the performance of both efficiency and responsiveness at the operational level. The benefits of the right CODP are raising delivery reliability, improving delivery speed, improving inventory cycle times, lowering logistics costs, lowering obsolescence risk, and improving product customization (Vank Hoek, 2000). Table 3-1 describes the different CODPs.

Table 3-1. Definition of the Customer Order Decoupling Points (Hayes, 2020)

Customer order decoupling point	Definition
Make-to-stock	Make-to-stock (MTS) is a traditional production strategy that is used by businesses to match the inventory with anticipated consumer demand. Instead of setting a production level and then attempting to sell goods, a company using MTS would estimate how many orders its products could generate, and then supply enough stock to meet those orders.
Assemble-to-order	Assemble-to-order (ATO) is a business production strategy where products that are ordered by customers are produced quickly and are customizable to a certain extent. It typically requires that the basic parts of the product are already manufactured but not yet assembled. Once an order is received, the parts are assembled quickly, and the final product is sent to the customer.
Make-to-order	Make-to-order (MTO) is a business production strategy that typically allows consumers to purchase products that are made to their specifications. It is a manufacturing process in which the production of an item begins only after a confirmed customer order is received.
Engineer-to-order	Engineer-to-order (ETO) is a business production strategy that typically allows consumers to purchase products that are engineered to their specifications. It

	is a manufacturing process in which the production of an item begins only after a confirmed customer order is received.

Figure 3-1 visualises the customer order decoupling points. Terberg is known as a market leader in the field of modifications of vehicles. A customer orders a vehicle to their own wishers. Therefore, Terberg is customer order-driven. They start producing existing configurations only when the customer confirms the order. Terberg has 2 different CODPs, MTO for existing configurations and ETO for new configurations. Terberg uses a customer order-driven strategy. This results in a high level of customisation but also longer lead times for the customers.

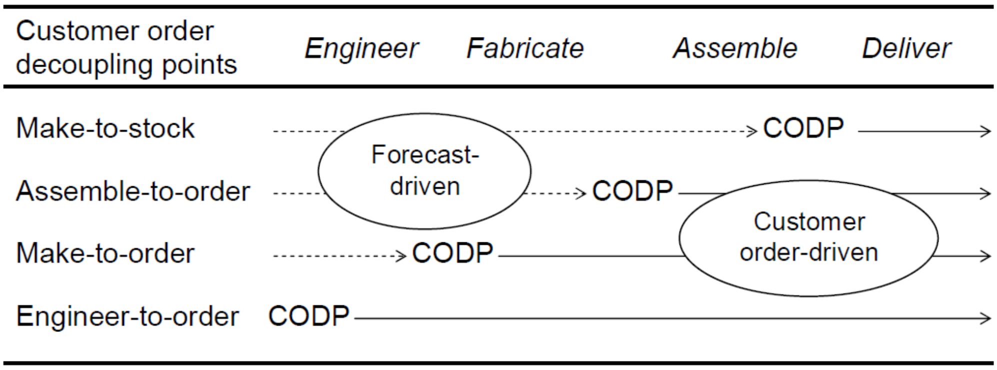


Figure 3-1. Different customer order decoupling points (Olhager, 2010)

3.1.2 Assembly Line Type

An assembly line is a flow-oriented production system where the productive units perform the operations. The workpieces visit stations successively as they move along the line. Originally, assembly lines were developed for a cost-efficient mass production of standardized products. However, the product requirements and the requirements of production systems changed dramatically. This resulted in different types of assembly lines (Boysen, Fliedner, & Scholl, 2006). Figure 3-2 shows different types of assembly lines.

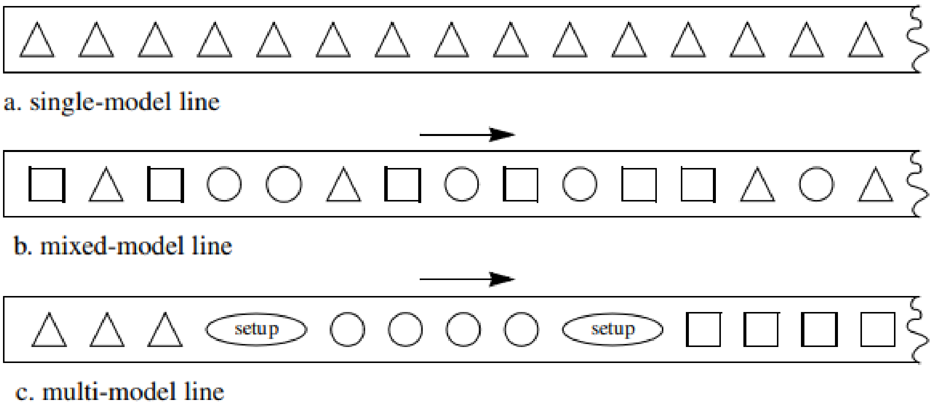


Figure 3-2. Assembly Lines for Single and Multiple Products (Becker & Scholl, 2004)

The single-model assembly (SMAL) line implies that one homogenous product is produced on the assembly line. This shifted to either a mixed-model assembly line (MMAL) or a multi-model assembly line (MuMAL). The MMAL is present in many industrial environments and produces several products on the same line in an intermixed sequence. The MMAL produces a mix of product types without setup types between different product types. This provides an assembly company with the opportunity maintain a high level of flexibility (Rabbani, Ziaeifar, & Manavizadeh, 2014). The article of Bukchin Dar-

El & Rubinovitz (2001) describes the MMAL in a MTO environment. Common characteristics of a MMAL in a MTO environment are a small number of workstations, lack of mechanical conveyance and highly skilled workers. This is also visible at Terberg. Terberg has a small number of workstations, no conveyance, and highly skilled workers.

The MuMAL requires a setup time between different product types. Therefore, MuMAL uses batch production However, both AS3 and AS4 have a stage with have setup times and benefit from batching products of the same type.

3.1.3 Supply Chain Planning Matrix

We classify our research problem with the use of the Supply Chain Planning Matrix (SCP-Matrix) of Meyer et al. (2008). The SCP-Matrix classifies the planning tasks into 2 dimensions, namely the *planning horizon* and *supply chain process*. The planning horizon is either long-term, mid-term or short-term. The supply chain process has the categories of procurement, production, distribution, and sales. Figure 3-3 shows the SCP-Matrix including both dimensions and the business processes per category. The long-term tasks are in a single box to illustrate the comprehensive character of strategic planning (Stadtler & Kilger, 2008).

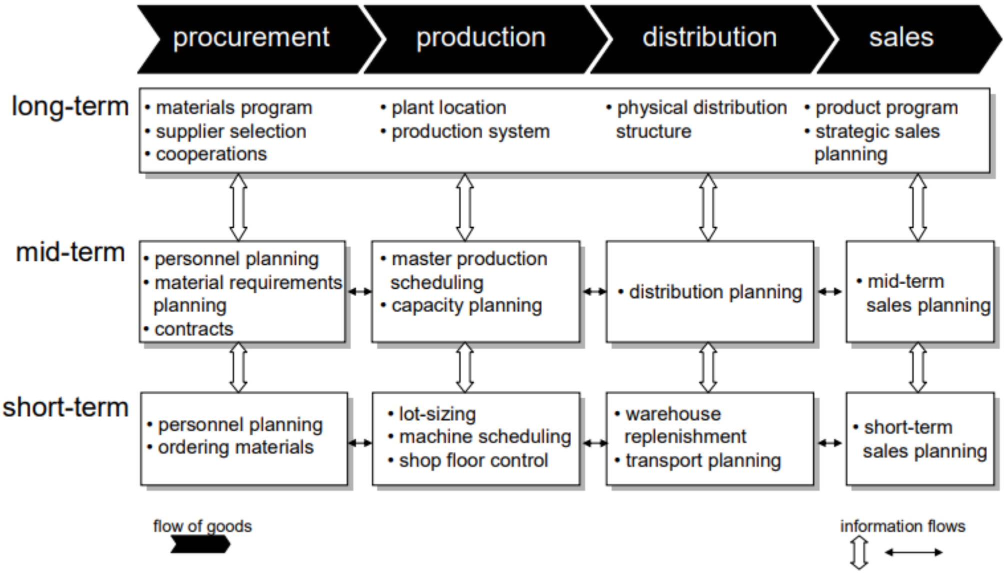


Figure 3-3. Supply Chain Planning Matrix (Meyer, Wagner & Rodhe, 2008)

Section 1.2.1 describes the research problem. It describes the Terberg has difficulties with creating a dynamic, predictive, and proactive assembly schedule that considers the variations and uncertainties. We locate our problem in the production category of the supply chain process dimension and in the short-term category of the planning dimension. Our research problem revolves around effectively scheduling the assembly process for each product, ensuring that it occurs on the appropriate machine and moment in time.

This category, where the problem resides, connects to other short-term categories and the mid-term production category. Section 1.2.1 highlights that Terberg encounters challenges when incorporating these categories into the creation of the assembly schedule. Additionally, proper integration between mid-term and short-term planning becomes increasingly necessary (Maravelias & Sung, 2009). There is a recent trend towards customization and diversification. This is also the case at Terberg. Terberg already has over 1000 options for customization and must deal with an increasing number of (electric) vehicle options.

3.2 Planning & Scheduling Model

This section introduces relevant planning and scheduling models for our research problem. Section 3.2.1. describes the car sequencing problem. Section 3.2.2. discusses the mixed-model assembly line. Section 3.2.3. analyses the flow shop scheduling problem and relevant extensions.

3.2.1 Car Sequence Problem

The car sequencing problem (CSP), developed by Parello et al. (2007), is a particular scheduling problem that has applications in managing assembly lines. The CSP has a list of vehicles which have options or variations that require higher work content and longer assembly times for at least one assembly station (Gravel, Gagné, & Price, 2005). Each station installs a different option and can only handle at most a certain percentage of cars passing along the assembly line. Cars are spaced such that the capacity of the station is never exceeded (Gottlieb, Puchta, & Solnon, 2003). A station handles only a maximum of r_i cars with option i from every s_i consecutive cars in the sequence (Kis, 2003). The capacity constraints of the station installing option i is given by $\frac{r_i}{s_i}$. Table 3-2 depicts an example instance of CSP. This example has 4 car types that can have 5 options. There is also a constriction about the maximum number of cars per consecutive car, e.g., 3 out of 5 consecutive cars can have option 1.

Table 3-2. Example Instance of CS with 4 Car types and 5 Options

Options	Car Types				Constraints
	1	2	3	4	r:s
1	X			X	3:5
2		X			1:3
3	X	X			2:5
4			X	X	2:3
5			X		1:4
Requested	2	2	2	2	

We see the CSP back at Terberg. More complex cars cannot be close to each other in the sequence. This results in too much workload on the assembly line for the fixed and flexible workers. However, the exact percentages are not known. There are also vehicle types that cannot go on the line rightly after each other. These vehicles are physically too long to go after each other.

3.2.2 Flow Shop Scheduling Problem

A flow shop scheduling problem (FSSP) relates to a production environment where jobs consisting of several products are processed in a predefined order. This predefined order is the same for all jobs. Each product requires a processing time and is assembled on a dedicated machine in the classical FSSP. The succeeding operation may only start after the current one finishes (Bultmann, Knust, & Waldherr, 2018).

Manufacturing companies emphasize scheduling products as close as possible to their due dates. The driving reason is the interest in Just-In-Time manufacturing. The new interest in scheduling is to analyse the impact on the manufacturing costs of earliness, i.e., producing products before due dates. The consequence of earliness is the costs occurring from finished goods inventory. The formulation of FSSP as an integer linear program of Chandra et al. (2004) is as follows:

Indices and index sets

i = index of jobs

j = index of machines

N = set of jobs, $\{i|i = 1,2, \dots., n\}$

S	=	set of machines, $\{j \mid j = 1,2, \ldots ., m\}$

Parameters

d_i	=	due date of job i
p_{ij}	=	processing time of job i on machine j

Variables

S_{ij}	=	start time of job i on machine j
C_{ij}	=	completion time of job i on machine j
T_i	=	tardiness of job i, $T_i = max(C_{im} - d_i, 0)$
E_i	=	earliness of job i, $E_i = \max(d_i - C_{im}, 0)$
y_{ik}	=	1, if job x is before job k in the sequence, $i, k \in N$ 0, otherwise

Mathematical model

$$\min Z = \sum_i E_i + T_i = \sum_i |C_{im} - d_i|$$

Subject to:

$$C_{ij} \geq C_{ij-1} + p_{ij} \tag{1}$$

$$S_{kj} - (S_{ij} + p_{ij}) + M(1 - y_{ik}) \geq 0 \qquad \forall\, i \in N, j \in S \tag{2}$$

$$S_{ij} - \left(S_{kj} + p_{kj}\right) + M\, y_{ik} \geq 0 \qquad \forall\, i, k \in N, j \in S \tag{3}$$

$$C_{im} - d_i = T_i - E_i \qquad \forall\, i, k \in N, j \in S \tag{4}$$

$$C_{ij} = S_{ij} + p_i \qquad \forall\, i \in N \tag{5}$$

$$C_{ij}, S_{ij}, E_i, T_i \geq 0 \qquad \forall\, i \in N, j \in S \tag{6}$$

$$y_{ik} \in \{0,1\} \tag{7}$$

The objective function of this model is to minimize total earliness and tardiness. The model has 7 constraints. Constraint 1 is the operation precedence constraint for a job. It ensures that an operation cannot start until the previous operation has been completed. Constraints 2 and 3 indicate job precedence at a machine. These constraints ensure that job i is scheduled before job k, and then at each machine job, k is started only after job i is completed. Constraint 4 determines E_i or T_i of a job. Constraint 5 determines the completion of each job at each machine. Constraint 6 ensures that variables C_{ij}, S_{ij}, E_i and T_i are positive values. Constraint 7 ensures that y_{ik} is binary.

Scheduling the jobs at Terberg adheres to the principles of the FSSP. The Planners schedule the jobs in a predetermined sequence on the machines. Terberg aims to finish jobs before their due date. However, this scheduling must not be too early due to the constraints of limited storage space available at the subsidiaries. The assembly lines of Terberg have additional variations, uncertainties, and restrictions. These extensions are elaborate in the next sections.

Flow Shop with Material Constraints

Manufacturing supply chains cope with non-renewable or consumable material. Shortages occur often in the current production environment. Therefore, the efficiency of an assembly process depends on the availability of required materials. Therefore, the scheduling process must consider the limited availability of materials. Laribi et al. (2016) describe the FSSP under resource constraints. Figure 3-3 illustrates an example of the FSSP under resource constraints. In the example, we assemble n jobs in m stages. There are R no renewable resources that jobs use in the stages. The inventory levels of the resources fluctuate over time. At certain moments in time the no renewable resources are resupplied, e.g., at t_1 and t_2 a resupply of no renewable resource l_1. An assembly only takes place when the required resources are present.

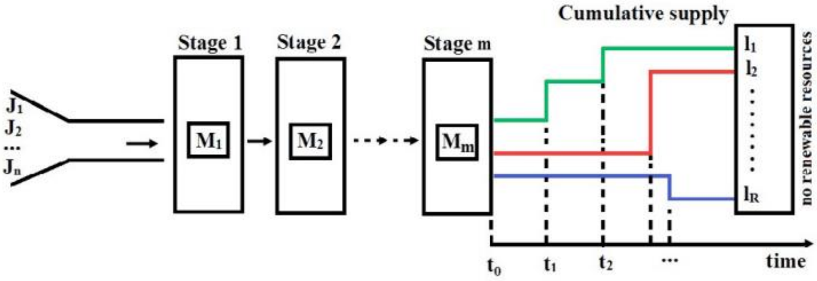


Figure 3-3 Problem Presentation Flow Shop Scheduling Under Resource Constraints (Laribi et al., 2016)

Terberg also deals with the fluctuation of spare parts. The inventory levels increase when a supply moment takes place and subsequently decrease when an assembly requiring that specific items take place. Additionally, there are some parts that block the assembly process. The assembly process can only continue when a certain part is available.

Flow Shop with Intermediate Buffers

Brucker et al. (2003) describe an extension of the classic FSSP with intermediate buffers. There are buffers of limited capacity between two consecutive machines. Once a job completes processing on a machine, it proceeds directly to the consecutive stage, or it is temporarily stored in the buffer between the stages.

In the context of Terberg, the available space between stages is restricted. There is limited capacity to store vehicles or vehicle systems during the assembly process.

Flow Shop with Pre-Assembly

Many manufacturing companies produce products by joining multiple components together. Each of these components needs to be processed before the assembly stage. The competitive market forces enterprises to have flexible production lines to produce a variety of products in large volumes and in the shortest time (Komaki, Sheikh, & Malakooti, 2018). Figure 3-4 shows a visual example of FSSP with manufacturing and assembly stations. The above example has a single machining stage, and all jobs have the same machining stage. The bottom stage is more complex. There are 2 machining and 2 assembly stages. Additionally, the jobs do not follow the same machining stages.

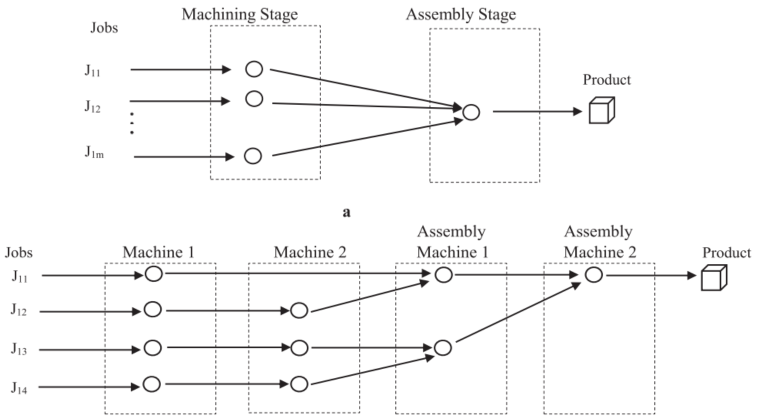


Figure 3-4. Visual Example of Flow Shop Combining Production and Assembly (Komaki & Malakooit, 2018)

Terberg also incorporates machining stations in its assembly processes. These pre-assembly stages must be finished before the main assembly stage can start. Additionally, we see that different products have different pre-assemblies.

Flow Shop with Parallel Machines

Nahhas et al. (2016) describe the hybrid FSSP, also known as the flexible FSSP. The hybrid FSSP has multiple parallel machines per operation instead of a single machine per operation. The number of parallel machines can differ per operation. The hybrid flow shop is a complex combinatorial problem encountered in many real-world applications (Ruiz & Rodríguez, 2010).

Terberg also has assembly lines with multiple workstations per stage. AS1, AS2 and AS3 are hybrid FSSP. All their assembly stations have multiple workstations to assemble multiple vehicles or vehicle systems at once. At AS1, the vehicles move simultaneously to the next stage. At the other 2 assembly lines, the vehicles move individually through the assembly line.

Hybrid Flow Shop with Dedicated Machines

Riane et al. (2010) present a hybrid FSSP with dedicated machines. Certain job types are only assembled on these dedicated machines. Manufacturing companies with a hybrid FSSP with dedicated machines use it to delay the production differentiation. Additionally, these companies use dedicated workstations to have a mixed-product assembly line where different products go through the same assembly stages. The jobs are processed on different machines depending on the specifications (Wang & Liu, 2012).

At Terberg we see the second approach. The jobs at AS2 and AS3 follow the same flow through the assembly line but are processed on different dedicated workstation in certain stages.

Hybrid Flow Shop with Worker Dependent Processing Times

Bultmann et al. (2018) present an FSSP with flexible processing times. In numerous flow shop variants, the processing times of the operations are not fixed and not exactly known in advance. The assembly times can differ per workstation at a certain workstation. This depends on the status of the machines or the number of employees working. Han et al. (2011) introduce the hybrid FSSP with worker constraint. In this problem each stage has multiple workers that operate at a certain workstation. These workers cannot work across multiple stages and/or specific workstations. Next to sequencing the jobs in each stage and assigning machines this scheduling problem also assigns the workers to corresponding workstation.

We see this problem also at all 4 assembly lines at Terberg. Each workstation needs workers to perform the tasks within the stage. Terberg assigns the workers to an operation and a specific workstation. The number of workers at a workstation influence the throughput time of a job at a workstation.

3.2.3 Scheduling Objectives

There are multiple objectives that indicate the performance of an assembly schedule. Ravndran et al. (2003) describe multiple interesting objectives for the scheduling problem. These potential scheduling objectives include:

- Maximum completion time
- Maximum flow time
- Maximum lateness
- Maximum earliness
- Total/average completion time
- Total/average weighted completion time
- Total/average flow time
- Total/average weighted flow time
- Total/average tardiness
- Total/average weighted tardiness
- Number of late jobs
- Total weighted number of late jobs
- Total/average earliness
- Total/average weighted earliness

Currently, Terberg exclusively focuses on the total throughput as the scheduling objective. However, in the interviews the employees highlighted several potential objectives. These objectives align with the objectives outlined in the existing literature.

Multi-Objective Function

It is difficult to suggest a schedule that optimizes all the performances together. Although one can construct an example for which a schedule may be good at one measure but perform poorly on others. The objective function for optimizing a scheduling problem with multiple objectives can be defined in various ways.

Table 3-3 presents the formulation of a multi-objective function as presented by Yenisey & Yagamhan (2013).

Table 3-3. Multi-objective Function Formulation (Yenisey & Yagmahan, 2013).

Notation	Explanation
Z	Single-objective problem where the aim is to minimize only Z
$f_w(Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where the aim is to minimize weighted k objectives (utility approach)
$f_p(Z_1 : Z_2 : \ldots : Z_k)$	Multi-objective problem where the aim is to minimize all objective (pareto-optimal approach)
$f_{np}(Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where the aim is to minimize all objectives, each objective is evaluated separately
$f_T\ (Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where the aim is to minimize the sum of the objectives $Z_1, Z_2, \ldots, Z_k$
$f_L(Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where the aim is to minimize a lexicographical order of all objectives, i.e., to minimize objective Z_1, then to minimize objective Z_2 subject to the optimality of objective Z_1, etc.
$f_\varepsilon(Z_p / Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where Z_p is the primary objective and other k objectives are subjected to bound constraints
$f_{gp}(Z_1, Z_2, \ldots, Z_k)$	Multi-objective problem where there are goals to reach for each objective in the problem

Terberg faces the challenge of balancing multiple objectives in their assembly process. The assembly line workers strive to complete all vehicles or vehicle systems in the shortest possible time to increase efficiency. On the other hand, the management aims to ensure that each product is finished on-time to avoid penalties, but also not too early to prevent potential storage or space issues at the assembly subsidiaries during post-assembly processes due to shortages. To address these objectives, Terberg considers all factors when determining the optimal assembly schedule.

Weights Decision Making

Ideally, weights of each objective function are assigned by the problem owner based on intrinsic knowledge of the problem. However, as different objective functions have different magnitude, the normalization of the objective values is required to get a pareto optimal solution consisting with the weights. The weights are computed as $w_i = u_i * \theta_i$, where u_i are the weights and θ_i the normalised factors. Normalization methods are:

- Normalize by the magnitude of the objective function at the initial point, $\theta_i = \frac{1}{f_i(x_0)}$;
- Normalize by the optimal value of the objectives, $\theta_i = \frac{1}{f_i(x^i)}$ where x^i solves $min_x\{f_i(x)\}$;

- Normalize by the difference of optimal function values in the worst and best points that give the length of the intervals where the optimal objective functions vary within the pareto optimal set, $0 \leq \frac{f_i(x)-z_i^{best}}{z_i^{worse}-z_i^{best}} \leq 1$.

The first two methods are proved to be ineffective and are not practical. The initial point may provide very poor representation of the function behaviour at optimality. Moreover, $f_i(x_0)$ is often equal to 0 and cannot be used at all. Use of the optimal solutions to individual problems can also lead to very distorted scaling since optimal values by themselves are in no way related to the geometry of the Pareto set (Grodzevich & Romanko, 2006).

3.3 Solution approaches

This section aims to identify suitable solution approaches for our scheduling problem. First, we categorize the different solution approaches available in the literature. Thereafter, we present promising solution approach for the problem.

According to Ruiz & Vázquez-Rodríguez (2010), the scheduling problem can be addressed using both exact and heuristic approaches. Further classification by Ribas et al. (2010) distinguishes heuristics into 2 main classes: constructive heuristics and improvement heuristics. It is worth noting that metaheuristics are a subset of improvement heuristics. Figure 3-5 illustrates the solution approach classification as described by Ruiz & Vázquez-Rodríguez (2010) and Ribas et al. (2010). This classification provides valuable insights into the different methodologies available for tackling scheduling problems, offering a range of techniques with varying levels of accuracy and computational complexity.

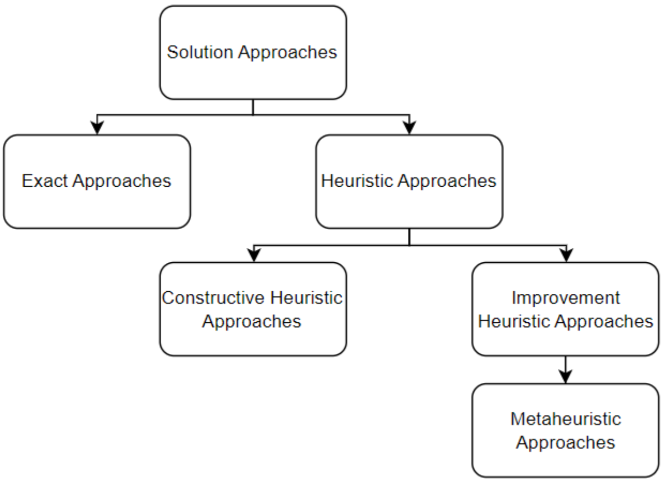


Figure 3-5. Solution Approach Classification.

The exact solution approaches provide an optimal solution. However, exact solution methods become inefficient for the large size problems. These problems are made of many jobs, stages, machines, and objectives. The exact approaches are still incapable to solve medium and large problem instances for real-world problems (Ruiz & Rodríguez, 2010). The balancing and sequencing problems are known as NP-hard combinatorial optimization problems (Seker, Özgürler, & Tanyas, 2013). Heuristic methods generate high-quality solutions in a reasonable time for practical use, but there is no guarantee of finding the optimal solution (Talbi, 2009). Since these problems cannot be solved in polynomial time, we focus on heuristic algorithms.

3.3.1 Constructive Heuristic Approaches

We utilize constructive heuristic methods to establish an initial feasible solution for the scheduling problem. These constructive heuristics are designed to efficiently plan activities or parts of activities by employing priority rules. By doing so, we aim to propose a constructive heuristic technique that can effectively generate an initial solution. This initial solution serves as a starting point for further optimization.

List Scheduling

List Scheduling is a non-hierarchical algorithm. It is non-hierarchical in the sense that, at each iteration, it selects an operation, assigns it to a machine, and determines a start time. This is in contrast with hierarchical methods that in a first phase assign operations to the workstation and in a second phase determine the starting times (Birgin, Ferreira, & Ronconi, 2014). If a tie exists, then usually the job is scheduled on the machine with the smallest index. Graham (1969) first introduces this method. The article of Wang & Cheng (1990) discusses a list scheduling algorithm for parallel tasks, the earliest completion time (ECT) algorithm. The algorithm takes the expected execution time also into account. The algorithm schedules the task on the machine with the lowest excepted completion time. Schutten (1996) proves that focusing on completion times rather than starting times results also in dominant list schedules for scheduling problems with dependent setup times.

Dispatching Rules

In addition, there are dispatching rules to determine the sequence of operations in a stage and on a workstation. The popularity of dispatching rules in practice is due to their low computational requirements. However, the results derived from a schedule constructed with a dispatching rule do not guarantee optimal or near optimal solutions (Linn & Zhang, 1999). Simple dispatching rules are First-In-First-Out (FIFO), Shortest Processing Time (SPT) and Earliest Due Date (EDD). Panwalkar and Iskander (1977) and Blackstone et al. (1982) classify and compare over a hundred dispatching rules. While these rules can handle dynamic problems and are easy to implement, they have a major disadvantage of being myopic. The decision is based on a situation with only a single machine (Tang, Liu, & Liu, 2005). Tang et al. (2005) introduces a neural network algorithm to solve a dynamic hybrid FSSP that is trained by these standard dispatching rules. Rolf et al. (2020) introduces a genetic algorithm to assign different dispatching rules instead of applying a standard single dispatching rule.

Nawaz-Enscore-Ham Algorithm

The MMAL and FSSP are combinational search problems with *n!* sequences, the sequence with the minimal total completion time could be identified. However, this procedure is quite expensive and impractical for large *n*. The article of Nawaz et al. (1983) describes a constructive algorithm to minimizes the makespan, called the Nawaz-Enscore-Ham (NEH) algorithm. The algorithm assumes that a job with more total process time on all the machines should be given higher priority than a job with less total process time. The step-by-step procedure is as follows:

Step 1. For each job i calculate

$$T_i = \sum_{j=1}^{m} t_{ij}$$

where t_{ij} is the process of job i on machine j.

Step 2. Arrange the jobs in descending order T_i.

Step 3. Pick the two jobs from the first and second position of the list of Step 2, and find the best sequence for these 2 jobs by calculating the makespan for

the 2 possible sequences. Do not change the relative positions of these 2 jobs with respect to each other in the remaining steps of the algorithm.
Set $i = 3$.

Step 4. Pick the job in the ith position of the list generated in Step 2 and find the best sequence by placing it at all possible i positions in the partial sequence found in the previous step, without changing the relative positions to each other of the already assigned jobs. The number of enumerations at this step equals i.

Step 5. If $n = i$, STOP, otherwise set $i = i + 1$ and go to Step 4.

Nagano & Moccellin (2002) introduce an extension to the NEH heuristic. The proposed heuristic penalizes the NEH job priorities T_i according to a lower bound for the total waiting time of a job. Lui et al. (2017) introduce another extension to the NEH heuristic This extension adds a tie-break rule and shows that performs slightly better than the classic NEH heuristic for some scenarios.

3.3.2 Improvement Heuristic Approaches

Once we establish an initial solution, we search for a better solution. With a metaheuristic we search for a better solution and improve objective function. Osman & Laport (1996) describe a metaheuristic as an iterative generation process that guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, in which learning strategies are used to structure information to find near-optimal solutions efficiently. There are many different metaheuristics. We discuss simulated annealing (SA), tabu search (TS) and variable neighbourhood search (VNS).

Simulated Annealing

Kirkpatrick, Gelatt and Vecchi developed in 1983 SA for solving NP-hard combinational and other optimization models to optimize the value of an objective function. The process of optimization in SA is to search for a solution (near) of global optimum. The process starts with an initial solution, a starting temperature T and a cooling factor α. This initial solution is random or is created with a constructive heuristic. The SA metaheuristic performs a stochastic search of the neighbourhood space (Hamzadayi & Yildiz, 2013). The neighbour solution is generated by using some operator that makes a small change in the current solution. Some examples of operators are 'swap', 'move', and 'insert'. The objective function of the new solution is compared to the current solution. If the solution is better, it replaces the current solution and if it is worse, it replaces the current solution by a probability which is obtained from the Boltzmann function $\exp\left(-\frac{\Delta}{\alpha * T}\right)$. This prevents the algorithm from being stuck in a local optimum. In this function Δ is the difference in objective function between the current solution and the new solution. The process of neighbourhood search continues until the number of iterations reaches to a predetermined value. After this step, the system temperature reduces. This process continues until the termination criteria is met (Hosseinabadi & Balas, 2016).

Tabu Search

Glover (1986) and Hansen (1986) introduced the idea of TS. The TS method is a metaheuristic which shares with SA algorithm the ability to avoid bad local optima. It uses a deterministic rather than stochastic acceptance criterion. Each iteration, TS moves to the best neighbour, even when the objective function is worse than the overall best. This may lead to returning to already visited solutions. To avoid this, a tabu list stores attributes of accepted solutions. The tabu list memorizes the recent search trajectory. A neighbouring solution is forbidden if it has attributes on the tabu list. Storing attributes rather than the complete solutions may cause non-tabu solutions to be wrongly prevented.

TS also avoids these cycles by discarding the neighbours that have been previously visited. In this way, the tabu list constitutes the short-term memory.

Variable Neighbourhood Search

VNS is a metaheuristic proposed by Mladenović and Hansen (1997). It represents a flexible framework for building heuristics for approximately solving combinatorial and non-linear continuous optimization problems. VNS systematically changes neighbourhood structures during the search for an optimal (or near-optimal) solution (Hansen, Mladenovic, Todosijević, & Hanafi, 2016). The VNS heuristic include an improvement phase to improve a given solution. This also use a shaking phase to resolve local minima traps. The improvement phase and the shaking procedure, together with neighbourhood change step (NCS) are executed alternately until fulfilling a stopping criterion.

The simple shaking procedure consists in selecting a random solution from the neighbourhood structure. In some cases, a complete random jump to a neighbourhood is too diversified. Sometimes it is preferable to do intensified shaking which considers how sensitive is the objective function to minor changes (shaking) of the solution.

The purpose of a NCS is to guide the VNS heuristic while exploring the solution space. It decides on which neighbourhood to explore and whether a solution is accepted as a new solution. The new solution replaces the initial solution if new solution is feasible and better (Shah-Hosseini, 2013). Sequential NCS, cyclic NCS, pipe NCS and skewed NCS are examples of NCS (Hansen, Mladenovic, Todosijević, & Hanafi, 2016).

As earlier mentioned, the improvement phase searches for the best solution within the neighbourhood. A local search heuristic is based on the exploration of a neighbourhood structure. Starting from an initial solution y, at each iteration it selects a better solution than y' for the neighbourhood structure. The local search finishes when it finds the local optimum in the neighbourhood (Hansen, Mladenovic, Todosijević, & Hanafi, 2016).

3.3.3 Neighbourhood Structures

This section elaborates upon neighbourhood structures available in the literature. A neighbourhood structure is a set of solutions that are considered neighbours of each other according to a specific criterion defined by the neighbourhood operator. It represents the collection of feasible solutions that can be reached from the current solution by applying the neighbourhood operator (Deng, Wang, Wang, & Zheng, 2016). The set of neighbourhood operators define the size of the neighbourhood. There are multiple operators to generate a neighbouring schedule for the (hybrid) FSSP. Al-Harkan et al. (2019) the following 6 operators for the hybrid FSSP:

1. (Randomly) swap 2 jobs on the same machine;
2. (Randomly) swap 2 jobs on different machines;
3. (Randomly) select 2 machines and swap the entire loading of jobs;
4. (Randomly) move a job on the same machine;
5. (Randomly) move a job to different machine;
6. (Randomly) move all jobs on all machines.

In addition to random job selection, a logical selection of jobs is also possible in the scheduling process. Zhang et al. (2018) consider the logical relationship between jobs and machine. The focus is on jobs and machines that have the potential to lead to improved schedules. The logical selection of jobs considers the number of jobs and the corresponding assembly time on a workstation. The neighbourhood operators select jobs from the fullest workstation to emptier workstations.

3.4 Chapter Conclusion

The goal of this chapter is to describe all relevant information to provide a clear answer to the following research question:

> *"What relevant information can we use from the literature to understand the problem at Terberg and formulate an effective solution approach to solve it?"*

Section 3.1 classifies the business strategy, assembly line and problem of Terberg into analytical frameworks from the literature to gain a deeper understanding of the problem. Terberg utilises both the MTO and ETO customer order decoupling points. The results in a high level of customisation but also leads to high lead times for customers (Olhager, 2010). Our research problem is in the short-term production category of the SCP-Matrix of Meyer et al. (2008). It is important to incorporate the adjecting categories (Maravelias & Sung, 2009).

Section 3.2 describes the relevant planning and scheduling models for our research problem. The CSP sseks to schedule vehicles with varying assembly requirements in a way that prevents overloading assembly stations (Gravel, Gagné, & Price, 2005). Terberg also experience the sequencing problem. Complex vehicles or vehicle systems cannot be next to or close to each other in the sequence. The FSSP involves the assembly of products in a predetermined order on machines (Bultmann, Knust, & Waldherr, 2018). Several extensions to the FSSP from the literature mirror the restrictions of 1 or multiple assembly lines within the scope of this research. Ravdran et al. (2003) introduces multiple objectives for the scheduling problem. It is challenging to suggest a schedule that optimizes all performance aspects simultaneously. Yenisey & Yagamhan (2013) presents multiple approaches to formulate a multi-objective function.

Section 3.3 introduces the solution approaches from the literature to solve a scheduling problem. Exact approaches are still incapable to solve medium and large problem instances for real-world problems (Ruiz & Rodríguez, 2010). Heuristic methods generate high-quality solutions in a reasonable time for practical use, but there is no guarantee of finding the optimal solution (Talbi, 2009). We utilize constructive heuristic methods to establish an initial feasible solution for the scheduling problem. List Scheduling is a non-hierarchical algorithm introduced by Graham (1969). The exact sequence of the jobs is determined by dispatching rules. Another promising constructive heuristic is the NEH algorithm introduced by Nawaz et al. (1983). There is a large variety in promising constructive heuristics. Commonly applied constructive heuristics are SA, tabu search and VNS. We create neighbourhood structures to generate new solution schedules. A neighbourhood structure is a set of solutions that are considered neighbours of each other according to a specific criterion defined by the neighbourhood operator (Deng, Wang, Wang, & Zheng, 2016). FSSP. Al-Harkan et al. (2019) introduces multiple operators for a scheduling problem. Zhang et al. (2018) consider operators with logical relationships between jobs and machine to improve schedules.

4 Model Description

The goal of this chapter is to answer the third research question:

> *"How can we systematically describe and model the scheduling problem of Terberg?".*

This chapter describes and models the present scheduling problem. Section 4.1 lists the assumptions and simplifications for the model. Section 4.2 classifies the assembly facilities. Section 4.3 presents the standard mathematical model including the indices, parameters, variables, restrictions, and objectives. This section also shows the solution decoding algorithm, the extensions to the standard model and the multi-objective function. Section 4.4 concludes this chapter.

4.1 Model Assumptions & Simplifications

To simplify the modelling of the problem, we assume the following:

- This model focusses only on the main assembly stage. For the pre-assemblies we only need to know when it needs to be ready for the corresponding main assembly stage. This is the start time of the corresponding main assembly stage;
- The buffers between the stages are infinite;
- When the assembly of a product is started, it does not stop until completed (no pre-emption);
- For each part, we know the initial inventory levels and the replenishment moments including the replenishment quantities. The sum of the initial inventory and the replenished quantities fulfil the total required quantity for each part;
- We allocate the parts based on the FCFS principles, even when a part is missing. We assign parts that become available through replenishment to the job that required the part first. This is even the case when we already finished the assembly without the missing part.

4.2 Assembly Line Model Classification

Section 2.1 describes the layout of the assembly lines within the scope of the research. We see that each facility is a version of the FSSP introduced in Section 3.2.2. Besides, we encounter that each assembly line also has different restrictions. Table 4-1 shows which extensions of the classic FSSP each assembly line contains.

Table 4-1. Assembly Line Characterization.

Assembly Line	AS1	AS2	AS3	AS4
Flow shop	X	X	X	X
Pre-assemblies	X	X	X	X
Due dates	X	X	X	X
Part usage with replenishment moments	X	X	X	X
Shortages dependent	X	X	X	X
Hybrid flow shops (parallel workstations)	X	X	X	
Workstation dependent assembly times		X	X	
Workstation-restricted jobs		X	X	
Synchronized cycle time movement	X (double)			X (single)
Sequence-restricted jobs	X			X
Setup times			X	X
Drying times			X	X

All 4 assembly lines are flow shops with pre-assemblies. The workers prepare sub-assemblies for the main assembly stages. In addition, the jobs of all assembly lines have due dates and need parts for assembly. These parts can be short during assembly. A shortage has impact on the (post-)assembly processes. Each shortage results in a later start time or extra work afterwards.

We see that AS1, AS2 and AS3 are hybrid FSSPs. Each stage in these assembly lines has multiple parallel workstations.

At AS2 and As3, the assembly time depend on the workstation where the assembly takes place. The number and skill level of the employees that are present at a workstation influence the assembly time. Besides, we have workstation-restricted jobs. At both assembly lines is not allowed to assemble these jobs on a different workstation.

We see that the AS1 and AS4 are FSSP with synchronized movement. AS1 moves 2 jobs in the simultaneously to the next stage after a fixed cycle time. AS4 moves a single job simultaneously to the next stage. Both assembly lines have a sequence-restricted jobs. Certain jobs cannot be placed rightly after each other on the assembly line.

Both AS3 and AS4 have stages where it is beneficial to sequence certain products rightly after each other. The workers prepare the assembly in certain stages less often since the setup of these products is similar. Both assembly lines also have a stage where the jobs dry after the operation. The jobs are ready at the start of the next working day.

4.3 Problem Modelling

This section defines the model of our scheduling problem. Section 4.3.1 presents the standard FSSP model that applies to each assembly line. It introduces the indexes, parameters, and variables of our FSSP. Section 4.3.2 explains the solution encoding and the decoding algorithm of the standard FSSP model. It describes the process of creating a schedule based on the sequences of jobs. Section 4.3.3 describes the assembly line specific extensions and modifications to the basic FSSP model. Additionally, it elaborates upon the effect on and adjustments to the solution encoding and/or decoding algorithm. Section 4.3.4 describes the objectives of the model.

4.3.1 Standard FSSP Model Notation

This section introduces the indices, parameters, and variables of the standard FSSP model. The standard model is the base for each assembly line. We extent the notation of the FSSP introduced in Section 3.2.2.

Indices

We use the following indices:

J	set of jobs
K	set of parts
I	set of stages
M_i	set of parallel workstations of stage i
P_{im}	set of serial positions in stage i on workstation m
T	set of time periods
j	index of jobs
k	index of parts
i, n	index of stages
m	index of workstations of stage i

p	index of positions in stage i on workstation m
t	index of time periods

We use several indices for the mathematical model. Set J represents all the jobs that need to be assembled, while set K contains all the parts that we use in the assembly. Set I represent the different stages in the assembly process. Set M_i contains the parallel workstations for a specific stage i. We use set P_{im} to indicate the position on workstation m in stage i. The maximum number of positions on workstation m equals the number of jobs that need stage i. Lastly, set T represents the time periods in the process.

Parameters

The model has the following parameters:

d_j	due date of job j
a_{jim}	assembly time of job j in stage i at workstation m
imp_k	impact level of part k
q_{kji}	quantity needed of part k for job j in stage i
r_{kt}	quantity replenished of part k at time t
inv_{k0}	initial inventory of part k at $t = 0$

As mentioned in Section 1.2.1 each $j \in J$ has a delivery date to the customer. Therefore, each job must be finished at a certain moment in time. Due date d_j represents the time that job j should be completed. There is a penalty when the due date is exceeded.

Section 1.2.1 describes that the assembly time differ per job and assembly stage. Besides, the assembly time depends on workstation the operation is assembled on. Assembly time a_{jim} is the assembly time of the operation of job j in stage i at workstation m.

Section 2.2.3 describes that each job consists of multiple parts. This section also mentions that each part has a different impact on the (post-)assembly processes when there is a shortage of it. The parameter imp_k presents the impact of part k. The impact value ranges from 1 to 4. The higher the value the more impact the part has on the additional post-assembly when this part is missing at the start of the assembly. An operation cannot start when a shortage occurs of a part with the impact value of 4. Parameter q_{kji} shows the quantity that job j needs of part k in stage i. There are also replenishment moments when parts are replenished. Parameter r_{kt} presents the replenished quantity of part k at time period t. Lastly, we know the initial inventory inv_{k0} of part k at the start of the scheduling period, when t equals 0.

Variables

The indices and parameters present the information that is available for the scheduling problem. We use this information to make certain decisions to create an assembly schedule. We express these decisions in the decisions variables. We make the following decisions when scheduling:

1. Determine the sequence of jobs on each workstation in each stage;
2. Determine the start- and end time each operation.

Regarding the first decision, we schedule each job *j* on a workstation in stage i. We determine in each stage which job is scheduled on which workstation and in which sequence we assemble these jobs. Regarding the second decision we need to determine the start time of these assemblies. We use a decoding algorithm to calculate the minimal start time of every assembly process on the allocated workstation. The start time of an assembly depends on the end time of the preceding assembly of that specific job, the preceding assembly on the specific workstation and the moment in time that all parts with impact value $imp_k = 4$ are available.

Based on both decisions we have the following variables:

o_{jimp}	1, if job j is processed in position p on workstation m in stage i 0, otherwise
s_{ji}	start time of job j on in stage i
f_{ji}	finishing time of job j in stage i
inv_{kt}	inventory level of part k at time t

We use several binary and continuous variables to model the problem. The binary variable o_{jimp} indicates if a job j is assembled in position p on workstation m in stage i. The variable s_{ji} and f_{ji} present the start- and finishing time of job j in stage i respectively. We use the decoding algorithm to determine the start- and end time of every assembly. Lastly, the variable inv_{kt} indicates the inventory level of part k at time period t. The inventory level changes over time due to the quantities used in the assemblies and the quantities supplied at the replenishment moments. We also use the decoding algorithm to calculate the inventory level at a certain moment in time. Section 4.3.2 describes the decoding algorithms to calculate the start- and end times and the inventory levels over time.

4.3.2 Solution Encoding and Decoding Algorithm

We use a solution encoding and a solution decoding algorithm to represent and solve our complex FSSP. The model in Section 4.3.1 introduces variable o_{jimp} to assign a job j to a specific position p on workstation m in stage i. This results in a sequence of jobs on each workstation in each stage. This sequence is the encoded solution. The solution decoding algorithm uses the encoded algorithm to generate an actual schedule. The actual schedule shows the start- and finishing times of each job in each stage.

Solution Encoding

We present the encoded solution of our FSSP by a sequence of jobs per stage and workstation. We denote the sequence of jobs on workstation m within stage i with vector V_{im}. By adopting the sequence per workstation approach, we effectively handle the constraints imposed by workstation-restricted jobs. This format allows us to easily determine which job is assigned to each workstation in every stage and verify if the job allocation is permissible. Additionally, given that our problem involves sequence-restricted jobs, it becomes crucial to establish the relationships between preceding and succeeding jobs. This encoding format enables clear identification of the jobs scheduled before and after a specific job, as well as verifying the feasibility of the job sequence.

Chapter 5 introduces the approaches to generate job sequences per workstation. The solution decode algorithm uses these sequences to generate a schedule. We move and swap jobs within and between sequences within a stage to explore alternative schedules.

Solution Decoding Algorithm

We use a solution decoding algorithm to obtain an actional schedule. We decode the encoded solution to obtain an actual schedule. The decode algorithm calculates the start- and finishing times. Additionally, we use the decoding algorithm to calculate the inventory levels and to check whether a part is missing at the start of an operation. With the decoded schedule we can evaluate the quality of the solution.

The start time of a job in a stage depends on the following 3 assembly prerequisites:

1. The finishing time of the assembly of the job in the preceding stage s_{ji}^{job};
2. The finishing time of the assembly of preceding job on the workstation $s_{ji}^{workstation}$;
3. The ready time of the critical parts for the job s_{ji}^{part}.

First, the maximum finishing time of the assembly of the job in the preceding stage s_{ji}^{job}. The job can continue when the assembly in stage $i-1$ is done. s_{ji}^{job} equals 0 in the first stage. Second, the finishing time of the job in the preceding position $p-1$ on the workstation $s_{ji}^{workstation}$. $s_{ji}^{workstation}$ equals 0 in the first position $p=1$ on each workstation m. Third, the ready time of critical parts s_{ji}^{part}. The operation starts when all parts with impact score $imp_k = 4$ are available for job j in stage i. We calculate the start time of the operation of job j in stage i as follows: $s_{ji} = max\left\{s_{ji}^{job}, s_{ji}^{workstation}, s_{ji}^{part}\right\}$.

The finishing time of an operation is calculated by adding the workstation specific assembly time a_{jim} to the start time, $f_{ji} = s_{ji} + a_{jim}$.

Additionally, we keep track of the inventory levels in the decoding algorithm. The initial inventory inv_{k0} is known for each part k. We know the needed quantity q_{kji} of each part k for job j in stage i. The needed quantities are subtracted from the inventory level inv_{kt} at time period $t = s_{ji}$. There are also moments in time that the inventory is replenished. We replenish the inventory of part k with r_{kt} at time period t.

We use an example situation to explain the decode. In the example situation, we have 3 jobs that have operations in 2 stages. Each stage has 2 workstations to schedule the assemblies on. Table 4-2 show the position of the jobs on the workstations in stage 1 and stage 2. Table 4-3 presents the assembly time per job, stage, and workstation.

Table 4-2. Example Vectors per Workstation of Stage 1 and 2

Stage i	**Workstation m**	**Vector V_{im}**
$i = 1$	$m = 1$	$V_{1,1} = \{1,3\}$
$i = 1$	$m = 2$	$V_{1,2} = \{2\}$
$i = 2$	$m = 1$	$V_{2,1} = \{1,2\}$
$i = 2$	$m = 2$	$V_{2,2} = \{3\}$

Table 4-3. Assembly time per Job, Stage and Workstation

$\boldsymbol{i = 1}$	$\boldsymbol{m = 1}$	$\boldsymbol{m = 2}$
$j = 1$	$a_{1,1,1} = 2$	$a_{1,1,2} = 1.5$
$j = 2$	$a_{2,1,1} = 9$	$a_{2,1,2} = 5$
$j = 3$	$a_{3,1,1} = 2$	$a_{3,1,2} = 1.5$

$\boldsymbol{i = 2}$	$\boldsymbol{m = 1}$	$\boldsymbol{m = 2}$
$j = 1$	$a_{1,2,1} = 4$	$a_{1,2,2} = 4$
$j = 2$	$a_{2,2,1} = 2$	$a_{2,2,2} = 2$
$j = 3$	$a_{3,2,1} = 3$	$a_{3,2,2} = 3$

The initial inventory levels of the 2 parts in the example are $inv_{1,0} = 4$ and $inv_{2,0} = 5$. Part 1 is an essential part, $imp_1 = 4$. This part is needed to start the assembly. Part 2 is a non-essential, $imp_2 = 3$. The assembly starts without this item. The inventory levels of the parts fluctuate over time. Table 4-4 present the needed quantities of part 1 and 2 respectively. Table 4-5 shows the replenishment moments of both parts.

Table 4-4. Bill of Materials of Part 1 and 2

$k = 1$	$i = 1$	$i = 2$
$j = 1$	$q_{1,1,1} = 1$	$q_{1,1,2} = 0$
$j = 2$	$q_{1,2,1} = 1$	$q_{1,2,2} = 0$
$j = 3$	$q_{1,3,1} = 1$	$q_{1,3,2} = 0$

$k = 2$	$i = 1$	$i = 2$
$j = 1$	$q_{2,1,1} = 0$	$q_{2,1,2} = 2$
$j = 2$	$q_{2,2,1} = 0$	$q_{2,2,2} = 3$
$j = 3$	$q_{2,3,1} = 0$	$q_{2,3,2} = 1$

Table 4-5. Replenishment Moments of the Part 1 and 2

Replenishment	$t = 5$	$t = 7$
Part $k = 1$	$r_{1,5} = 2$	$r_{1,7} = 0$
Part $k = 2$	$r_{2,5} = 0$	$r_{2,7} = 3$

The decoding algorithm starts with calculating the start- and end times of the first operations in the sequence in the first stage. The first stage does not have a preceding stage. s_{ji}^{job} equals 0 the operations in this stage. The first operations in the sequences do not have a preceding job. $s_{ji}^{workstation}$ equals 0 for these operation in the first positions. s_{ji}^{part} also equals 0 since the needed quantities are available. Therefore, $s_{1,1} = 0$ and $s_{2,1} = 0$. The inventory level of part 2 drops from 4 to 2, since $q_{1,1,1} = 1$ and $q_{1,2,1} = 1$. Based on the start- and assembly time we calculate the finishing times, $f_{1,1} = 2$ and $f_{2,1} = 5$. We continue with the assembly of job 3 in stage 1. There are still enough items of part 1 present, $inv_{1,2} = 2$. Workstation 2 assembles job 3 in stage 1 at $s_{3,1} = 2$, since $s_{3,1} \geq max \left\{s_{s_{3,1}}^{job} = 0, s_{s_{3,1}}^{workstation} = 2, s_{s_{3,1}}^{part} = 0\right\}$.

In stage 2, we assemble job 1 at $s_{1,2} = 2$, since $s_{1,2} \geq max \left\{s_{1,2}^{job} = 2, s_{1,2}^{workstation} = 0, s_{1,2}^{part} = 0\right\}$. Workstation 2 has job 3 at the start of the sequence. The assembly starts at $s_{32} = 4$, since $s_{32} \geq max \left\{s_{3,2}^{job} = 2, s_{3,2}^{workstation} = 0, s_{3,2}^{part} = 0\right\}$. $inv_{2,4} = 0$ after the operation of job 3 in stage 2. The operation of job 2 in stage 2 can only start after the replenishment of part 2, $r_{2,7} = 3$. We assemble job 2 in stage 2 at $s_{2,2} = 7$, since $s_{2,2} \geq max \left\{s_{2,2}^{job} = 4, s_{2,2}^{workstation} = 6, s_{2,2}^{part} = 7\right\}$. All start- and finishing times result in a schedule. Figure 4-1 shows the Gantt chart of the schedule that results from the solution decoding algorithm.

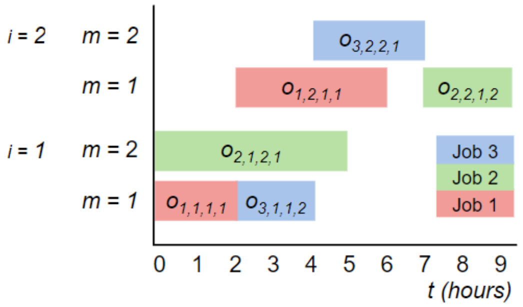


Figure 4-1. Gantt Chart of the Schedule

Additionally, we have insight in the fluctuation of the inventory levels of the parts. Figure 4-2 shows the inventory levels of part 1 and 2 over time.

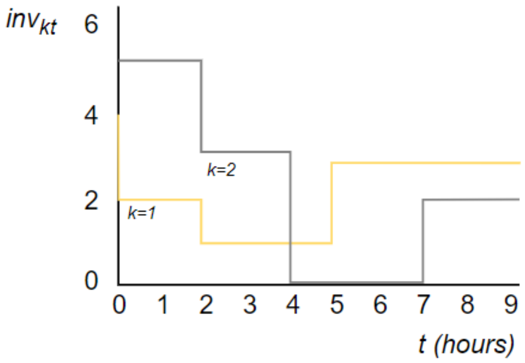


Figure 4-2. Inventory Level Over Time

The solution decoding algorithm updates the start- and end times, as well as the inventory levels, continuously. This results in an actual schedule. We use the actual schedule created by the solution decoding algorithm to calculate the objective values. Section 4.3.3 shows the objectives.

4.3.3 FSSP Model Assembly Line Extensions

Table 4-1 shows the characterizations of each assembly line. These characterizations have influence on the modelling of the problem. This section presents the extensions to the standard FSSP model described in Section 4.3.1. Additionally, it specifies the assembly lines to which the extension is applicable, and it explains the modifications to the decoding algorithm.

Workstation-Restricted Jobs

AS2 and AS3 have job types that we cannot assemble at every workstation. Therefore, we introduce the following parameter:

x_{jim} 1, if job j can be processed on workstation m in stage i
0, otherwise

The assembly $o_{jim_ip_{im}}$ can only take place if job j can be processed on workstation m_i in stage i. In the encoded solution, we only allow the solution approach to schedule jobs in a position on workstations where the operation can take place. The solution decoding algorithm does not change.

Sequence-Restricted Jobs

The AS1 and AS4 have jobs that cannot be assembled rightly after each other in a stage. We know if each job j can succeed job q in stage i on the same workstation m_i. We introduce the following parameter:

z_{jqim} 1, if the assembly of job j can succeed the assembly of job q in stage i
0, otherwise

The assembly of job j in stage i on workstation m in position p_{im} with job q in the preceding position $p_{im} - 1$ is only allowed when $z_{jwim} = 1$. In the encoded solution, we only allow the solution approach to schedule jobs in a position on workstations where the operation can take place. The solution decoding algorithm does not change.

Setup Time

The AS3 and AS4 have stage with setup time. The setup time depend on the preceding job that is on a workstation in a stage. We express the setup time in the following parameter:

v_{jqi} setup time for job j in stage i when job q is the preceding job

The setup time influences one of the factors that determine the start time of a job. The workers start assembling job j on workstation m in stage i when they finish job w in the preceding position and have the setup ready for job j. This results in a different $s_{ji}^{workstation}$.

We apply the setup time v_{jqi} when job q is in the preceding position p_{im} in relation to job j in stage i on the same workstation m. In the solution decoding algorithm, we update the calculation of $s_{ji}^{workstation}$.

Drying Time

The AS3 and AS4 have at least a single stage with drying time. The jobs move to the next stage at the start of the next working day. We introduce the following parameter:

y_i	1, if stage i has drying time 0, otherwise

When a preceding stage has drying time, the s_{ji}^{job} of the next stage is rounded up to the start time of the next working day. For example, when a working day has 8 hours and a job in a stage with drying time finishes at $t = 3$. The next stage can start at least at $t = 8$, therefore $s_{ji}^{job} = 8$. In the solution decoding algorithm, we update the calculation of s_{ji}^{job}.

Synchronized Cycle Time Movement

The jobs of AS1 and AS4 move synchronized to their next stage after a cycle time. The introduce the following parameter:

c	cycle time of in the assembly line

Each main stage of AS1 has 2 workstations. The jobs on both workstations move to the workstations in the next stage simultaneously. The workers on both workstations within a stage work together to assemble the 2 parallel jobs. Each main of AS$ has a single workstation. In this case, the assembly of each job j takes place on the only workstation m in the same position p_{im} in each stage i. We schedule jobs in the same position on the workstation(s) in each stage. This does not change the solution decoding algorithm.

Section 2.1.1 mentions that the workload differs for the assembly of each job in each stage at AS1. However, the jobs move in these assembly to the next stage after a certain cycle time. We forbid scheduling a job in a position when the average workload of the operations in the same position in a stage exceeds the cycle time. For example, $a_{ji1} + a_{ki2} \leq c * 2$, in stage i the sum of the assembly times of job j and k in both workstations cannot exceed the cycle time. We multiple the cycle time with 2 since we have 2 workstations.

4.3.4 Objectives and Multi-Objective Function

This section elaborates on the objectives and the objective function of the scheduling problem. Section 2.3.3 describes that currently Terberg only uses the assembly quantity per week as a performance indicator. However, as indicated by the involved employees of Terberg there are more interesting indicators that present the quality of an assembly schedule.

According to the literature review and the input of employees related to the assembly, planning and scheduling process, there are other suitable objectives for the scheduling problem. Section 3.2.3describes interesting objectives for the scheduling problem and multiple ways to deal with multi-objective scheduling problems. We use the following 8 objectives to evaluate the quality of a schedule:

Total Makespan

Section 1.2.2 describes that Terberg aims to produce a certain number of jobs in the least amount of time. We introduce the makespan as a scheduling indicator to represent this objective. Next to maximizing the throughput it also maximizes the utilization (Yenisey & Yagmahan, 2013). Equation 4-1 presents the calculation of the makespan objective.

Equation 4-1. Makespan Objective

$$C_{max} = max_{i\in I, j\in J}\{f_{ji}\}$$

last finishing time $max_{i\in I, j\in J}\{f_{ji}\}$ of all the assemblies taking place is the makespan of the entire sequence. We aim to minimize this objective.

Average Makespan

Section 2.1 describes that there is limited buffer space available. Terberg aims to minimize the individual makespan to minimize the time that a job is placed in a buffer. We calculate the average makespan with Equation 4-2.

Equation 4-2. Average Makespan Objective

$$C_{avr} = \frac{\sum_{j=1}^{J} max_{i\in I}\{f_{ji}\} - min_{i\in I}\{s_{ji}\}}{J}$$

We calculate the makespan of a single job j by subtracting the earliest start time $min_{i\in I}\{s_{ji}\}$ from the latest finishing time $max_{i\in I}\{f_{ji}\}$. The average makespan is the sum of all single makespan divided by the total number of jobs J.

Number of Jobs On-Time

Section 1.2.2 describes that penalties are attached to each job. The goal is to schedule the job in such a way that it is finished before we reach the predetermined due date d_j. We aim to maximize the jobs that are on-time. We calculate the number of jobs that are assembled on-time with Equation 4-3.

Equation 4-3. Jobs On-Time Objective

$$J_{ot} = \sum_{j=1}^{J} D_j, \quad D_j = \begin{cases} 1, & (d_j - max_{i\in I}\{f_{ji}\}) \geq 0 \\ 0, & otherwise \end{cases}$$

We introduce the auxiliary variable D_j to determine if a single job is finished on-time. D_j equals 1 when the job is on-time and 0 when it is not. A job j is on-time when the due d_j is lower than the latest finishing time $max_{i\in I}\{f_{ji}\}$ of a job. We aim to maximize this objective value. $C_{max}C_{avr}$

Number of Jobs Finished without Shortages

Section 1.2.1 also describes that Terberg deals with shortages. Terberg aims to maximize the number of jobs that finish without shortages. The workers need extra time to assemble the missing parts after the assembly. We calculate the number of jobs that finish the assembly line without missing parts with Equation 4-4.

Equation 4-4. Jobs No Shortages

$$J_{ns} = \sum_{j=1}^{J} R_j, \quad R_j = \begin{cases} 1, & \sum_{i=1}^{I}\sum_{k=1}^{K} max\left\{0, q_{kji} - inv_{k,s_{ji}}\right\} \geq 0 \\ 0, & otherwise \end{cases}$$

We introduce auxiliary variable R_j to determine if a single job is assembled without missing parts. $R_j$equals 1 if there are no parts missing and 0 when there are. An operation that starts with missing parts when the inventory inv_{kt} of part k at time s_{ji} is less than the quantity needed q_{kji}. We aim to maximize this objective value.

Number of Jobs On-Time and No Shortages

We combine the 2 objectives above into a new objective. It is even more beneficial to finish a job on-time without missing parts than finishing a job with missing parts. Therefore, we aim maximize the number of jobs that are both finished on-time and do not miss parts. We calculate this objective value with Equation 4-5.

Equation 4-5. Jobs On-Time without Shortages Finished Objective

$$J_{ot+ns} = \sum_{j=1}^{J} \max\{0, D_j + R_j - 1\}$$

We take both the earlier mentioned auxiliary variables D_j and R_j. A job is finished on-time without shortages when both variables equal 1. We aim to maximize this objective indicator.

Hours Too Late

It is a hard challenge to ensure that all jobs are assembled on-time. Therefore, we also aim to minimize the hours that the jobs are too late. Equation 4-6 calculates the total hours that the jobs are assembled too late.

Equation 4-6. Hours Too Late Objective

$$H_{tl} = \sum_{j=1}^{J} max\, \{0, max_{i \in I}\{f_{ji}\} - d_j\}$$

The number of hours too late of job j equals the difference between the latest finishing time $max_{i \in I}\{f_{ji}\}$ and the due date d_j, when the due date is before the latest finishing time. We aim to minimize this objective value.

Weighted Number of Shortages

Another objective is the total number of shortages that are present when the jobs are assembled. We aim minimize the weighted number of shortages since each part has a difference impact on the post-assembly time. Calculate the objective with Equation 4-7.

Equation 4-7. Weighted Number Shortages Objective

$$N_s = \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{k=1}^{K} max \left\{0, q_{kji} - inv_{k,\, s_{ji}}\right\} * imp_k$$

We compare the quantity needed q_{kji} of job i in stage j with the inventory inv_{kt} at the start of the operation s_{ji}. There is a shortage when there are not enough items present to fulfill the demand. We correct the number of shortages with the impact score.

Weighted Hours Until Replenishment of Shortages

The replenishment moments take place during the assembly period. As earlier mentioned, we assign a part that becomes available through replenishment to the assembled job that required the part first. A job gets the part when we fulfil the demand of all earlier scheduled jobs that require that part. We also use the weighted factor for this objective. Equation 4-8 calculates the time weighted hours till supply.

Equation 4-8. Weighted Hours till Supply of Shortages Objective

$$H_{ts} = \sum_{j=1}^{J} \sum_{i=1}^{I} \sum_{k=1}^{K} (t'_k - s_{ji}) * imp_k$$

In the equation the t'_k presents the moment in time that total replenished quantity can fulfil the needed quantity of part k. The replenishment moment t'_k of part k is the first time that the total replenished quantity $\sum_{t=s_{ji}}^{t'_k} r_{kt'}$ can fulfil the demand of the inventory level $inv_{k,s_{ji}}$ and the required quantity q_{kji} of part k used by the assemble of job j in stage i starting at s_{ji}. We aim to minimize the time till replenishment to reduce the amount of time a job remains unfinished.

Multi-Objective Function

We have multiple objectives that determine the quality of a schedule. There are a total of 8 performance indicators. According to the literature review in Section 3.2.3, there are serval multi-objective functions f. We deal with a significant difference in importance of the performance indicators. We use the lexicographical order optimization. This approach optimizes objective Z_1, then objective Z_2 subject to the optimality of objective Z_1 etc. We have the objective function $f_L(C_{max}, C_{avr}, J_{ot+ns}, J_{ot}, J_{ns}, H_{tl}, N_s, H_{ts})$ which optimizes the objectives in lexicographical order.

4.4 Chapter Conclusion

This section concludes the chapter. The goal of this chapter is to describe all relevant information to provide a clear answer to the following research question:

"How can we systematically describe and model the scheduling problem of Terberg?".

Section 4.1 presents a list of assumptions and simplifications for our scheduling problem. Section 4.2 characterizes the scheduling problem at Terberg as a hybrid FSSP. We see that each assembly subsidiary has its own extensions and restrictions. These extensions and restrictions are also found in the literature review. Table 4-1 classifies each assembly line according to the extensions and restrictions.

Section 4.3 elaborates upon the modelling of the problem. First, we introduce the standard FSSP model notation. These indices, parameters and variables are the base model for each assembly line within the scope of our research. Second, we describe the solution encoding and decoding algorithm. We present the encoded solution as a sequence of jobs on the workstations in each stage. The decoding algorithm calculates the start- and end times of each operation based on the encoded solution. It also calculates the inventory levels over time. Third, we describe the extensions of standard FSSP model and to which assembly subsidiaries these extensions apply. We have the following extensions:

- Workstation-restricted jobs
- Sequence restricted jobs
- Setup times
- Drying times
- Synchronized cycle time movement

Lastly, we determine the objectives and the multi-objective function. We have a total of 8 objectives. We use the lexicographical order optimization. This approach optimizes objective Z_1, then objective Z_2 subject to the optimality of objective Z_1 etc. We deal with a significant difference in importance of the performance indicators. Therefore, we use this hierarchical optimization approach. This is based on a hierarchical We have the objective function $f_L(C_{max}, C_{avr}, J_{ot+ns}, J_{ot}, J_{ns}, H_{tl}, N_s, H_{ts})$ which optimizes the objectives in lexicographical order. We determine the lexicographical order together with Terberg.

5 Solution Design

The goal of this chapter is to answer the fourth research question:

"Which alternative solutions approaches are suitable to solve the scheduling problem of Terberg?"

This chapter discusses the best alternatives to solve the specific FSSP that Chapter 4 describes. It also motivates why we select these approaches. The general idea of this chapter is to develop an approach that finds the sequences of jobs that result in a suboptimal schedule per assembly line. First, we utilize a constructive heuristic to logically create a schedule from scratch. Thereafter, we use improvement heuristics to improve the constructive schedule. Section 5.1 elaborates upon the constructive heuristic approaches to create an initial solution schedule. Section 5.2 describes the improvement heuristic approaches based on local search to create an improved solution schedule. Section 5.3 concludes the chapter. Appendix B contains flow charts of all constructive and improvement heuristics.

5.1 Constructive Heuristic Approaches

This section discusses the design of our constructive solution approaches of our FSSP. We use 4 approaches to construct a schedule. Chapter 4 describes that the FSSP involves multiple jobs, stages, workstations, and parts. Additionally, each assembly line has its own restrictions. This makes the scheduling problem complex. To find the optimal solution is computationally infeasible. We use constructive heuristics to create suboptimal schedules while being achievable within an acceptable timeframe.

Section 5.1.1 describes a random construction heuristic for benchmarking the performance of the improvement heuristics. Section 5.1.2 describes the constructive NEH heuristic that Nawaz at al. (1983) propose. According to Ruiz et al. (2008), it is vastly superior to other constructive heuristics for small and medium-sized versions of the hybrid FSSP. The assembly line within our scope belong to these versions of the hybrid FSSP. One of the key strengths of the NEH heuristic is its ability to continuously evaluate the schedules that it generates, this allows the algorithm to choose the best option at each step. This iterative process of the NEH heuristic ultimately produces a suboptimal schedule.

Section 5.1.3 describes list scheduling. It is an efficient heuristic to generate a feasible schedule for the parallel workstation scheduling problem. Since our problem has multiple parallel workstations in series the heuristic is also applicable. One of the key benefits of the list scheduling heuristic is its efficiency in generating schedules within a relatively short amount of time. The heuristic prioritises jobs and workstations to logically create a schedule from scratch. Section 5.1.4 introduces an approach based on the NEH heuristic that uses the job prioritisation of the list scheduling approach to exclude scheduling positions to save computational time.

5.1.1 Random Construction Heuristic

The Random Construction Heuristic (RCH) is a method that creates an initial solution by randomly assigning jobs to workstations. This approach is useful for benchmarking the performance of the other constructive heuristic in producing a high-quality schedule.

Process Explanation

The RCH starts with selecting a random operation of a random job. We assign this operation to a random position in the sequence on a random workstation in the corresponding stage. Then, we select another job with at least 1 unscheduled operation. We assign this job to a random position in sequence on a random workstation. We continue this process until we scheduled all jobs on a workstation in all stages. We consider the workstation-restricted job constraint and the sequence-restricted job constraint. We exclude these infeasible positions in de encoded solution. We only schedule the operation in a position in the sequence that satisfy the constraints. This results in a sequence of jobs on each workstation in each stage. This is the input for the solution decoding algorithm.

5.1.2 NEH Heuristic

Section 3.3.1 introduces the Nawaz-Enscore-Ham (NEH) heuristic. The NEH is a constructive heuristic introduced by Nawaz et al. (1983). According to Table 4-1, each of the assembly lines within the scope of our research are FSSP, of which 3 hybrid FSSP. The NEH heuristic is a vastly superior constructive heuristic compared to other constructive heuristics for the hybrid FSSP (Ruiz, Serifoglu, & Urlings, 2008). The heuristic evaluates the objective values during scheduling, this ensures the creation of a suboptimal schedule.

We adapt the NEH heuristic to suit our FSSP. The standard NEH heuristic prioritizes jobs based on their influence on the overall assembly time of the assembly line. The heuristic schedules the jobs in order of decreasing influence. The job's influence is determined by its total assembly time (Yong, Zhantao, Xiang, & Chenfeng, 2022). AS2 and AS3 deal with workstation-restricted jobs. We also consider the number of workstations where we can assemble the jobs on besides prioritising the jobs by their tot assembly time. We schedule these workstation-restricted jobs first to give them a position in the sequence on these specific workstations. This ensures that these positions are not taken by non-workstation-restricted jobs.

Process Explanation

The NEH heuristic consists of the following steps. First, we count the number of stages where a job needs a specific workstation. We place these highly workstation-restricted jobs on top of the list to reserve a spot on these workstations before other jobs take these positions. Section 4.3.3 introduces parameter x_{jim}. This parameter shows if job j is schedulable on workstation m in stage i. We use auxiliary parameter X_{ji} to indicate if job j needs a specific workstation in stage i. $X_{ji} = 1$ when the total number of workstations M_i does not equal the number of workstation where we can schedule job j. Equation 5-1 presents the calculation of the number of workstation-restricted stages (WRS_j) for each job j.

Equation 5-1. Number of Workstation-Restricted Stages per Job

$$WRS_j = \sum_{i=1}^{I} X_i \ where \ X_i = \begin{cases} 1, & \sum_{m=1}^{M_i} x_{jim} < M_i \\ \quad 0, & otherwise \end{cases}$$

We use the total average duration of the assembly time as a second priority criterion. We only consider the assembly times of the workstation where the job is allowed to be assembled, when parameter x_{jim} equals 1. Equation 5-2 presents the calculation of the total average assembly time ($TAAT_j$) for each job j.

Equation 5-2. Total Average Assembly Time per Job

$$TAAT_j = \sum_{i=1}^{I} \frac{\sum_{m=1}^{M_i} a_{jim} * x_{jim}}{\sum_{m=1}^{M_i} x_{jim}}$$

We use both factors to create the NEH priority list. First, we prioritise the jobs based on the highest number of workstation-restricted stages. We prioritise jobs with the highest total average assembly time when the number of workstation-restricted stages is equal. The jobs at the top of the list are jobs with the least number of schedulable workstations and the highest assembly time. On the bottom of the list are the jobs with the most schedulable workstation and the lowest assembly time.

We continue the NEH heuristic by scheduling the first job on every position in the first stage. We use the solution decoding algorithm of Section 4.3.2 to create the schedule and calculate the objective values of Section 4.3.4. We schedule the job in the position that results in the best objective value. We continue to the next stage and schedule the first job on every position in this stage. Again, we evaluate the objectives values and schedule the job in the best position. We continue this process till the job is scheduled in each stage. We delete the job from the NEH priority list and select the new job on top of the list. We schedule the job in every position in the first stage without changing the job that is already scheduled. We continue this process for every stage till the job is scheduled in each stage. This iterative process repeats itself till all jobs have a position in all stages. We consider the workstation-restricted job constraint and the sequence-restricted job constraint. We exclude these infeasible positions in de encoded solution. We only schedule the operation in a position in the sequence that satisfy the constraints.

The NEH algorithm has a worst-case time complexity of $O(n^3 \log n)$, where n is the number of jobs to be scheduled. The sorting of the jobs based on the impact on the overall assembly time has a complexity of $O(n \log n)$. The iterative insertion step has a time complexity of $O(n^2)$.The NEH heuristic is a time-consuming algorithm for large instances. The algorithm checks every possible position within a sequence. The total number of positions is the sum of the allowed workstations and the number of jobs that are already scheduled on these workstations. Therefore, we elaborate an adjusted version of the NEH heuristic in Section 5.1.4. We combine it with the prioritisation aspects of list scheduling to exclude certain scheduling positions. Section 5.1.3 explains the prioritisation of list scheduling.

5.1.3 List Scheduling

Section 3.3.1 introduces the list scheduling (LS) proposed by Graham (1969). LS is a greedy algorithm for parallel machine scheduling. The LS algorithm is a (time) efficient heuristic to generate a feasible schedule for the single parallel machine scheduling problem. The heuristic considers the priority of jobs and workstations. This ensures that we create a suboptimal schedule. Table 4-1 shows that 3 assembly lines within the scope have parallel workstations. However, our scheduling problem deals with multiple stages in series with parallel workstations. We perform the LS algorithm multiple times. We execute the LS algorithm the same number of times equal to the number of stages in the assembly line.

The input to LS is a list of jobs that need assembly in the first stage. The job list is sequenced by the job priority rule. We schedule the first job on the list on one of the workstations according to the workstation priority rule. We schedule each job on one of the workstations till all jobs are scheduled. In our problem, we continue to the next stage and prioritise the jobs for the second time. Again, we schedule all jobs on one of the workstations based on the workstation priority rule. This process continues till all jobs are scheduled on a workstation in each stage. Jobs can overtake each other in the hybrid flow shops. We re-prioritise the jobs per stage since the relative arrival time changes for each job.

We use the aspects of the jobs to create the job priority list. Table 4-1 shows that the jobs in each assembly line have due dates and need parts. These parts have replenishment dates. These parts also have a different impact on the (post-)assembly time. Our version of the LS algorithm uses these due dates, availability of parts, replenishment dates and impact of shortages to prioritise jobs in the schedule. Additionally, we use the departure time of a job to determine the priority in the succeeding stage. The earlier a job is released from the previous stage, the more priority it gets. Therefore, the priority list of a stage has influence on the next stage. We include the end time of a job in the job priority in the stages with a preceding stage.

The other aspect of LS is selecting a workstation for a job. Wang & Cheng (1990) discuss a dominant list scheduling algorithm for parallel workstations. The LS algorithm schedules jobs on the workstation with the expected earliest completion time. Table 4-1 shows that AS2 and AS3 deal with workstation-restricted jobs. We prioritize the workstations where majority of the jobs can be assembled. This frees up space for the workstation-restricted jobs that come after.

Process Explanation

Our LS algorithm starts with a priority list for the first stage in the assembly line. Section 4.3.1 describes the parameters. We use the parameters related to the jobs and the needed parts to determine the priority value of the jobs. First, we introduce the earliest weighted replenishment date priority rule to determine the in which sequence we want to schedule the jobs. The weighted replenishment date considers the initial inventory level inv_{k0}, required quantity q_{kji}, replenishment moment t, replenishment quantities r_{kt} and the impact imp_k of part k. The priority of scheduling a job early is high when the initial inventory is high, the required quantity is small, the replenishment moment is early, the replenishment quantity is high, and the impact is low of a missing part. Equation 5-3 presents the calculation of the weighted replenishment date per job j (WRD_j). Our primary priority rule is the earliest weighted replenishment date (EWRD).

Equation 5-3. Weighted Replenishment Date

$$WRD_j = \sum_{i=1}^{I} \sum_{k=1}^{K} q_{kji} * \frac{\sum_{t=0}^{T} r_{kt} * t}{\sum_{t=0}^{T} r_{kt}} * imp_k$$

We consider the earliest due date (EDD) of the jobs as the secondary priority rule when the WRD_j is equal for multiple jobs. We schedule the job with a lower due date in the beginning of the schedule. Both job priority rules result in a single job priority list.

Next, we assign each job to a workstation m in the first stage. We schedule the job on the workstation with the earliest finishing time (EFT). If multiple workstations have the same earliest expected finishing time for a particular job, we prioritize scheduling the job on workstation m that has the highest number of schedulable jobs (NSJ) at the workstation. Parameter $x_{jim} = 1$ when a job is schedulable on workstation m in stage i. Equation 5-4 presents the calculation of the number of schedulable jobs per workstation m (NSJ_m).

Equation 5-4. Number of Schedulable Jobs on a Workstation

$$NSJ_m = \sum_{j=1}^{J} x_{jim}$$

We continue to the next stage when we schedule all jobs on a workstation in the first stage. Again, we prioritize the jobs. In this priority list we also consider the release time from the first stage. At first, the priority of job j is determined by the earliest release date (ERD) priority rule. We take the finish time $f_{j,(i-1),m}$ of the preceding stage $i-1$. The earlier the release time the more priority. When the release time of multiple jobs is equal, we use the EDD as the second job priority rule. This process continues till we schedule each job on a workstation in every stage.

List scheduling may not always produce an optimal solution, especially for complex instances with large variations in assembly time and restricted scheduling positions. Therefore, other optimization algorithms and heuristics are used in combination with list scheduling to improve the quality and efficiency of the scheduling algorithm. Section 5.1.4 presents list scheduling in combination with the NEH heuristic of Section 5.1.2.

5.1.4 Adjusted NEH with Position Exclusion

As mentioned in Section 5.1.2, the traditional NEH scheduling algorithm is time consuming for large instances. Section 2.2.2 describes that Terberg aims to generate a schedule of multiple weeks instead of 1 week. This increases the number of jobs we need to schedule. The more jobs we schedule, the greater the number of potential positions available to sequence the jobs in a stage. Section 2.3.3 mentions that Terberg wants an approach that creates a sub-optimal schedule within reasonable time.

The computational time increases when the instance size increases. The algorithm evaluates an increasing number of positions each time we schedule a job. We use the job priority aspect of the LS algorithm to get an indication of the best position to schedule the job. We can eliminate positions that are highly unlikely to produce a favourable schedule. For example, a job with a high due date and a low weighted release date, compared to the jobs that already scheduled on the workstation, is probably placed in any of the first positions on a workstation. In this case, we do not evaluate the lost positions on the workstation.

Figure 5-1 visually displays the disparity in the number of jobs to scheduling between the standard NEH algorithm and the adjusted NEH scheduling algorithm. As more jobs are added to the schedule, the greater the number of positions we eliminate. We save running time by eliminating positions. The reduction of running time depend on the number of jobs we exclude.

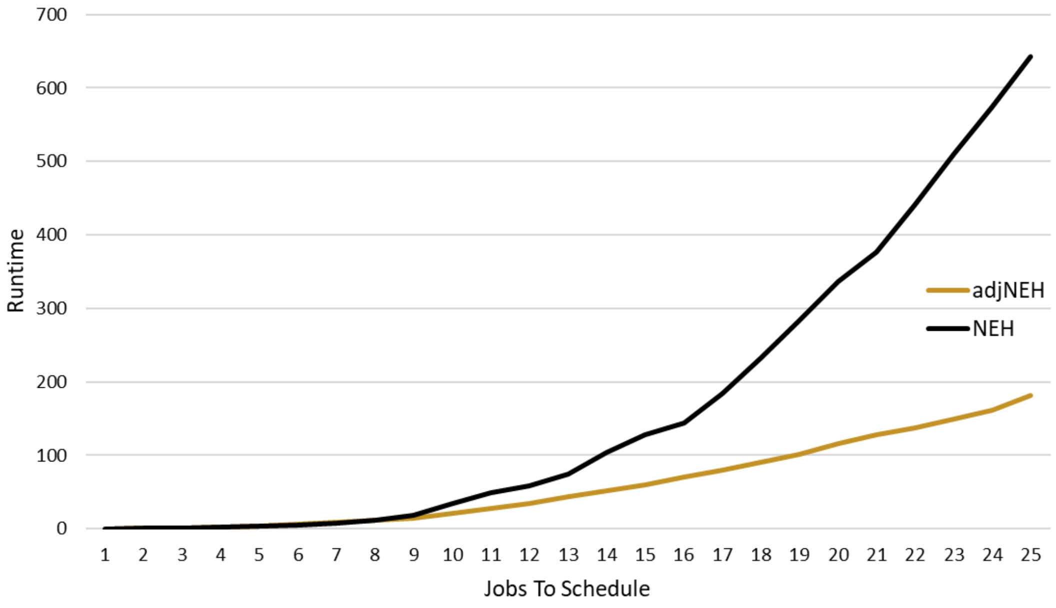


Figure 5-1. Runtime To Schedule Jobs

Process Explanation

The adjusted NEH with position exclusion starts the same as NEH heuristic in Section 5.1.2. We create a NEH priority list. This priority list is based on the number of schedulable workstations and the total assembly time per job. This is the order in which we schedule the jobs. When we schedule a job on a workstation that already has a job, we deviate from the standard NEH procedure. Specifically, we consider scheduling the job only in its designated position and a predetermined number of adjacent positions.

Section 5.1.3 mentions 3 job priority rules, EWRD, EDD and ERD. It depends on the stage we are scheduling which primary and secondary rule we use. We construct a job priority list based on the primary and secondary priority rule. We evaluate the priority values of the jobs on the workstation and the job we want to schedule. The values give an indication in which position in the sequence we potentially schedule the job.

Different from the normal NEH, we consider only schedule the job in this position and a pre-determined number of positions around this position, the evaluation radius. The evaluation radius equals the number of positions we want to evaluate around the position in the evaluate each position and schedule the job into the position with the best objective value. We repeat this process until all jobs are scheduled on a workstation in each stage.

This approach introduces 1 input variable. This is the number of positions we evaluate around the expected position that corresponds to the priority list position. This variable is the evaluation radius.

5.2 Improvement Heuristic Approaches

This section describes the improvement heuristic approaches. With the improvement heuristic we improve the constructed schedule. Section 5.2.1 introduces the neighbourhood structures. We use operators to generate new neighbourhood solutions. Section 5.2.2 describes a simple improvement heuristic (SIH) to benchmark the performance of the other improvement heuristic. We prefer SIH over a steepest descent improvement heuristic since our hybrid FSSP has a high level of complexity. Our problem involves multiple jobs, stages, and workstations. This results in a large neighbourhood structure. Our constructive heuristics prioritise jobs and workstation to generate a sub-optimal schedule. These constructive heuristic use aspects of jobs and workstations Additionally, the constructive heuristics evaluate the objectives continues during the construction of the schedule.

Section 5.2.3 focusses on simulated annealing (SA). According to Naderi et al. (2009), SA shows promising results for the hybrid FSSP. We prefer SA over other improvement heuristics since it effectively analysis neighbourhood schedules. The SA parameters allow us to influence the process of exploration, exploitation and the running time.

5.2.1 Neighbourhood Structures

Section 3.3.3 introduces multiple neighbourhood operators to create new solution spaces. The SIH and SA use these operators to create new neighbourhood structures. We have 4 operators to create neighbour solutions. Swap and move operators are commonly used for the hybrid FSSP and result in promising improvements (Engin, Ceran, & Yilmaz, 2011). We use the following operators:

Operator	Description
Single Swap	Swap the positions of 2 jobs in a single stage
Full Swap	Swap the positions of 2 jobs in every stage
Single Move	Move a job to a different position in a single stage
Full Move	Move a job to a different position in all stages

Each operator has its benefits in the creation of the neighbourhood structure. We state the advantages per operator in each corresponding section below. We provide an example per operator for more clarity.

Single Swap

The single swap neighbourhood operator intensifies the search within the solution space. This operator allows us to swap 2 jobs within a single stage of the assembly line while keeping the number of jobs per workstation unchanged. The single swap aims to find a better solution by making small adjustments to the current schedule. Additionally, we do not influence the schedule in previous stages. Figure 5-2 shows an example of a single swap.

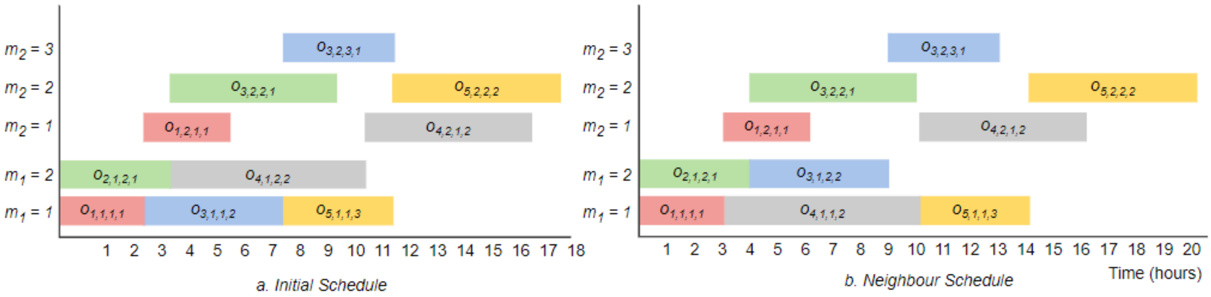


Figure 5-2. Single Swap Example

In the example, assembly $o_{3,1,1,2}$ (depicted in blue) and $o_{4,1,2,2}$ (depicted in grey) in swap stage 1. The positions in the sequences in stage 2 do not change. The single swap increases the makespan from 18 to 20.

Full Swap

The full swap neighbourhood operator diversifies the search in the solution space. We swap the positions of 2 jobs in all stages of the assembly line. In contrast to the single swap operator, the full swap operator explores the global search space by considering the flow and interactions between different stages in the assembly line. Figure 5-3 shows an example of a full swap.

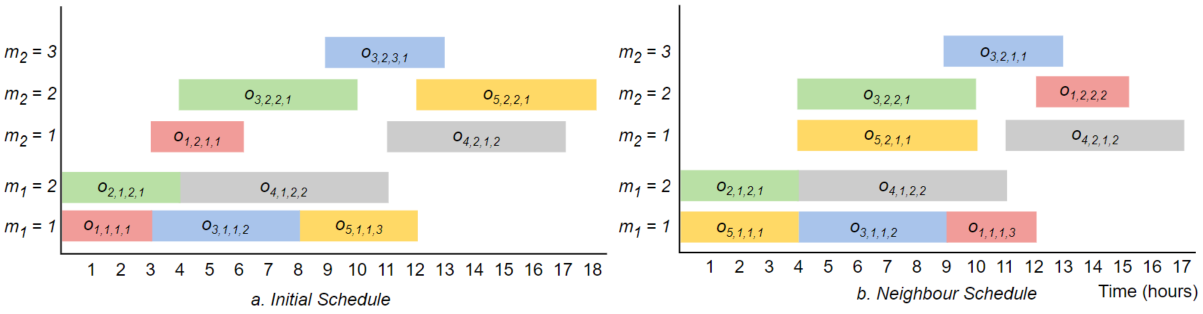


Figure 5-3. Full Move Example

The example shows that job 1 (depicted in red) job 5 (depicted in yellow) swap positions in both stage 1 and 2. The full swap reduces the makespan from 18 to 17.

Single Move

The single move neighbourhood operator also intensifies the solution space. The operator moves a job to a different position in a single stage. With a move we change the number of jobs per workstations and insert a job in a sequence. By only moving a job in a single stage we do not influence the sequence of jobs in other stages. Figure 5-4 shows a move of a single move.

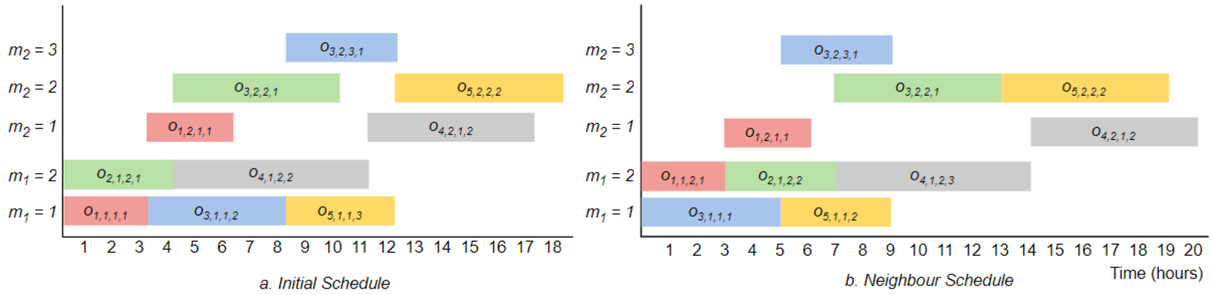


Figure 5-4. Single Move Example

Operation $o_{1,1,1,1}$ (depicted in red) moves from the first position on the first workstation to the first position on the second workstation and becomes operation $o_{1,1,2,1}$. All assemblies in stage 2 do not change from positions. The single move operator increases the makespan from 18 to 20.

Full Move

The full move neighbourhood operator diversifies the search in the solution space. We move a job to a different position in all stages. In contrast to the single move, the full move operator explores the goalball search by considering the flow and interaction between different stages in the assembly line. Figure 5-5 shows an example of a full move.

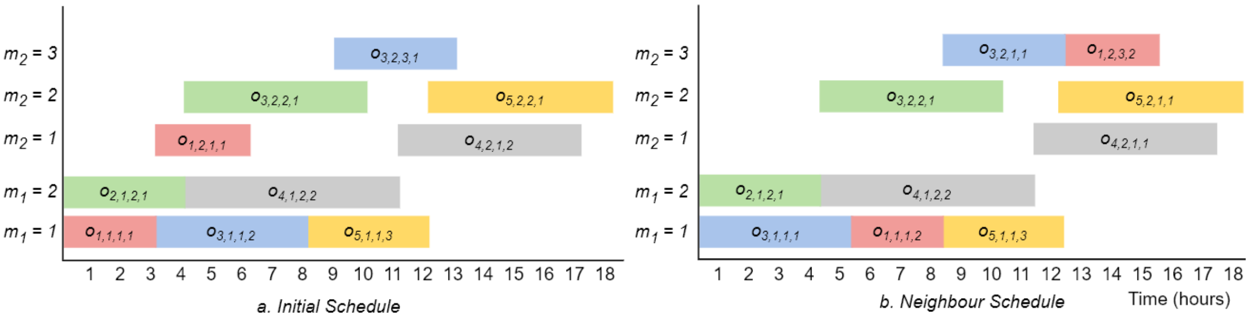


Figure 5-5. Full Move Example

The example shows that that job 1 (depicted in red) moves in the first stage to the second position on workstation 1 and in the second stage to the second position on workstation 3. The full move operator does not change the makespan.

We can create numerous possible schedules with these neighbourhood operators, which may not all result in better schedules. Zhang et al. (2018) presents that logic behind operators result in better solutions. Therefore, we also create logic behind the selection of neighbourhood operators. We only allow swaps and move that potentially generate better schedules. Experiment 2 examines the behaviour of the 4 neighbourhood operators. In this experiment we look for patterns in the selection of jobs and positions in the schedule.

5.2.2 Simple Improvement Heuristic

The first improvement heuristic is the simple improvement heuristic (SIH). The SIH only accepts a solution if the objective value is better. The SIH cannot escape local optima. We use the performance of the SIH to benchmark the performance of the more advanced improvement heuristic, that can escape local optima.

The SIH starts with an initial solution S. This solution becomes best solution S^*. The SIH generates a neighbour solution S' from S^*. We accept S' as the new best solution if the objective of the neighbour solution $F(S')$ is better than the objective of the best solution $F(S^*)$. This process continues until we reach the time limit t_{limit}.

Recall the examples of Section 5.2.1. The SIH algorithm only accepts a neighbour solution when the objective, the makespan, decreases. Therefore, the algorithm only accepts the second example.

5.2.3 Simulated Annealing

Section 3.3.2 describes the Simulated Annealing (SA) heuristic developed by Kirkpatrick, Gelatt and Vecchi (1983). We use SA for our FSSP to efficiently search for a global optimum. The SA algorithm accepts the neighbour solution S' when it is better than the current solution S. SA includes randomness to prevent the algorithm from being stuck in a local optimum. In contrast to SIH, the algorithm accepts a solution even when the resulting objective value is worse.

The algorithm accepts the neighbour solution with a probability that depends on the objective value of neighbour $F(S')$ and current solution $F(S)$ and the progression of the heuristic. We denote the progression of the heuristic by temperature T. The algorithm reduces temperature T over time with cooling factor α. Therefore, the acceptance probability decreases over time such that it is less likely that SA accepts worse solutions over time.

SA continues until the number of iterations equals a predetermined value. We call these iterations the Markov Chain Length M. After this step, the system temperature reduces. The SA algorithm continues until the termination criterion is met (Hosseinabadi & Balas, 2016). We use a minimal temperature T_{stop} to stop the heuristic. The heuristic stops when $T \leq T_{stop}$. We also have a starting temperature T_{start}. In the end, SA returns the best solution S^*. All in all, SA has 4 parameters, T_{start}, T_{stop}, α, and M.

Each parameter influences the process of exploration, exploitation, and the running time. The T_{start} influences the exploration and exploitation process. A higher T_{start} encourages greater exploration. The T_{start} does not directly impact the computational time but affects the balance between exploration and exploitation, which can influence the convergence speed. The same applies to the T_{end}. However, a higher T_{end} prioritises exploration over exploitation. The M also influences the exploration and exploitation process. A longer M increase the exploration capability of SA and results in a longer running time as the algorithm takes more iterations to explore the solution space. The α directly impact the running time. A higher and slower factor leads to a higher computational time as SA explores more extensively. This also accelerates the exploration process of SA.

Recall the examples of Section 5.2.1. The SA algorithm always accepts the second example since the objective, the makespan, decreases. Other than SIH, there is a chance that we also accept the generated neighbourhood schedules in the first, third and fourth example.

5.3 Chapter Conclusion

This section concludes this chapter. The goal of this chapter is to describe all relevant information to provide a clear answer to the following research question:

"Which alternative solutions approaches are suitable to solve the scheduling problem of Terberg?".

To answer this question, we use heuristic approaches to solve the hybrid FSSP of Terberg. The problem is too complex to use an exact approach. We use local search to find the best schedule. Section 5.1 describes 4 constructive heuristics to create an initial schedule. We need an initial schedule to start the local search. The first constructive heuristic generates a random initial solution. We use this heuristic to benchmark the improvement heuristic. Second, we introduce the NEH constructive heuristic. NEH evaluates the generated schedules constantly and chooses the best. Third, we use a version of list scheduling to create an initial schedule. LS is a time efficient heuristic that prioritises job and workstations. Fourth, we use an adjusted version of the NEH heuristic that excludes scheduling positions based on the prioritisation aspect of list scheduling. The exclusion of positions reduces the computational time.

Section 5.2 describes the neighbourhood operators and the improvement heuristics. We consider 4 neighbourhood operators: the single swap, full swap, single move, and full move. We consider 2 improvement heuristics. First, a simple improvement heuristic that does not accept worse solutions to benchmark the performance of the other improvement heuristic. Second, we use simulated annealing to improve the schedule. Simulated annealing accepts worse solution to escape a local optimum.

6 Experiments & Results

The goal of this chapter is to answer the fifth research question:

"Which alternative solutions approach performs best compared to each other and how does the best solutions approach perform under different experimental settings at Terberg?"

This chapter describes the problem instances we use to test the solution approaches and the conceptual experiments that we perform to evaluate the performance of the solution approaches. Section 6.1 describes the problem instances of the assembly lines that we use for the experiments. Section 6.2 elaborates upon the experimental design. It explains per experiment which and how many instances we use to perform the experiments. It also shows the results of these experiments and presents the conclusions per experiment. Section 6.3 concludes this chapter.

6.1 Problem Instances

This section focuses on the creation of problem instances and provides a summary of the process. Additionally, it provides statistics of the problem instances per assembly line. We use the up-to-date assembly data of 2022 as the base to create the problem instances. Some data lacks completeness and reliability. For example, historical inventory levels are not accessible, or the assembly times of certain jobs are incorrect. The generated problem instances closely resemble the real-life instances and are validated by Terberg. The next section explain how we create the instances in more detail. The assembly data of AS4 is not available.

We categorize the input parameters in 3 parts, assembly line parameters, job parameters, and part parameters. This division allows for a more organized and comprehensive representation of the input data. We have the following parameters:

1. Assembly line parameters:
    a. Number of stages;
    b. Number of workstations per stage;
    c. Number of jobs (per job type):
    d. Assembly line specific extensions:
        i. Job-restricted workstations per stage;
        ii. blocked consecutive jobs per stage;
        iii. setup times per stage;
        iv. drying time per stage.
2. Job parameters:
    a. Assembly time per stage and workstation;
    b. Due date;
    c. BOM;
3. Part parameters:
    a. Initial inventory level;
    b. Replenishment moments;
    c. Replenishment quantities per moment.

Section 6.1.1 describe the assembly line parameters. Section 6.1.2 explain the job parameters and Section 6.1.3 explain the part parameters. We use AS2 as an example in each section. The parameters for the remaining assembly lines are obtained in a similar manner.

6.1.1 Assembly Line Parameters

Regarding the first set of parameters, Section 2.1 presents the layout of each considered subsidiary. This includes the number of stages and workstations. The number of stages and workstations are the same in each problem instance per assembly line.

Section 2.2.2 shows that each assembly line creates a schedule per week. Therefore, there is a limit of jobs we schedule per week. Besides, there is a limited number of jobs per job type we schedule per week. When we exceed this number, the schedule is infeasible or inefficient. In this case the workload in a single stage is too high or a job-restricted workstation is over-scheduled. From the historic data we see the percentage of assembled jobs per job type. Table 6-1 shows an example of the distribution of assembled jobs per job type.

Table 6-1. Distribution of Assembled Jobs per Job Type

Job Type	Type1	Type2	Type3	Type4	Type5	Type6	Type7	Type8	Type9
Percentage Assembled	0.277	0.07	0.03	0.223	0.34	0.024	0.012	0.022	0.003

We create instances based on these distributions. However, we still consider the maximum number of jobs per job type to create correct instances. This ensures that we do not create unrealistic instances.

Additionally, we incorporate assembly line-specific extensions into the problem instances. We know per stage which jobs are restricted to certain workstations and which job sequences are blocked on the workstation. We also know the setup and drying time of the stages per job.

6.1.2 Job Parameters

Regarding the second set of parameters, we know the assembly time per stage and workstation for each job. We know the assembly time of each job per stage and workstation. This is the same in each problem instance.

The due dates of jobs in real-world scenarios vary significantly. To assess the optimization capabilities of the solution approaches we narrow the range of due dates. We decrease the range of due dates to allow a more precise evaluation. This identifies the strengths of the solution approaches. The jobs' due dates follow a uniform distribution $U_{[a,b]}$, with the lower bound a being the average assembly time rounded up to the nearest day. We round this value up to nearest day. The upper bound b is equal to the sum of the lower bound and the time period we want to schedule. When the scheduling period is 1 week (5 working days), the upper bound $b = a + 5$.

Each job can require a different set of parts. This also applies to jobs within the same job type. Based on historical data, we can determine the number of items needed for each job type.

Table 6-2 provides an example. The example shows the required items per job type. This includes their impact, quantity, the stage in which they are needed, and the percentage of occurrence per job type.

Table 6-2. BOM Information

Job Type	**Item ID**	**Impact**	**Quantity**	**Stage ID**	**Occurrence (%)**
Type1	Item1	4	1	55	100.00
Type1	Item2	2	2	70	53.23
Type1	Item3	2	1	55	74.93
Type2	Item3	2	1	55	100.00

Type2	Item4	1	1	60	95.45

We use the percentage of occurrence to generate the BOM for the problem instances. We generate a list of parts we need per job and list with the total number of items we need per part. We use this information to determine the initial inventory, supply dates and quantities.

6.1.3 Part Parameters

The third parameter category is related to the parts. Based on the BOM of each job we calculate the total number of items we need per part. The inventory levels are not available in historic data. Therefore, the initial inventory levels are a random value between 0 and the total number of items needed per part. We determine the supply dates and quantities with the use of historical data. We assume that both are normally distributed $N(\mu, \sigma^2)$.

The first supply moment of a part is always a random day within the average scheduling horizon. This ensures that there are (essential) parts available in the first part of the week. We calculate the next supply moments with the average and standard deviation of the replenishment time. We also have data available regarding the supply quantities. We know the average and standard deviation of the supply quantities. We need as much supply moments till we fulfil the total demanded number of items per part. Table 6-3 shows the supply data of a couple of parts.

Table 6-3. Supply Data of Parts

Item ID	Average Replenishment Time (Days)	Std. Replenishment Time (Days)	Average Supply Quantity (Items)	Std. Supply Quantity (Items)
Item1	1.29	0.70	5.76	2.84
Item2	2.28	1.03	1.25	0.71
Item3	10	0	56	33.94
Item4	15	0	4	0
Item5	5	0	500	0

We use both the replenishment time and supply quantity distributions to generate supply moments for our instances. We ensure that the total quantity of required items equals the sum of the items in the initial inventory and the total quantity of items replenished.

Table 6-4 shows the important parameters per assembly line. It shows statistics related to the number of workstations per stage, total assembly time, number of jobs and parts. Additionally, it shows the assembly line specific restrictions per assembly line.

Table 6-4. Instance Statistics per Assembly Line

Assembly Line	AS1	AS2	AS3
Workstations (per stage)	Constant	Varying	Varying
Assembly Time (per stage)	Constant	Varying	Varying
Number of Jobs (per week)	Medium	Low	High
Parts (per job)	Medium	High	Low
Dedicated Workstation	-	Yes	Yes
Blocked Sequences	Yes	-	-
Setup Time	-	-	Yes
Drying Time	-	-	Yes

**Depends on the cycle time*

Each assembly line has a different assembly layout. AS1 has the same number of workstations per stage. The jobs move to the next workstation after a pre-determined cycle-time. We use a cycle time for SA1. Both AS2 and AS3 assembly line have a varying number of workstations per stage. The assembly time also differs per stage for these assembly lines. There is relatively more deviation in the total assembly time at AS2. The number of jobs assembled per week also differs per assembly line. The AS1 assembles a medium number of jobs, AS2 assembles a low number of jobs, and the AS3 assembles a high number of jobs per week. Both AS1 and AS2 assembly complex products, whereas AS2 produces the most complex vehicles. This is also visible in the total number of parts per job. A AS2 vehicle needs the most parts, and AS3 product requires the least.

The assembly line specific restrictions differ per assembly line. Both the AS2 and AS3 have dedicated workstations. AS2 has high number of the jobs need a least 1 dedicated workstation. Whereas AS2 has a low number of jobs that need at least 1 dedicated workstation. AS1 only has blocked sequences, this is a low percentage. Only AS3 has setup times and drying time. A low-medium number of the jobs have setup times in the second stage. All the jobs of AS3 need drying time in this second stage.

6.2 Experiments Design & Results

This section provides an overview of the experimental design and the results. The experimental design consists of 2 phases, the setup experiments and the alternative models experiments. We use the setup experiments to initialise the solution approaches and identify the best solution approach. We use the model alternative experiments to evaluate the performance of the best solution approaches under different circumstances. We perform a total of 6 experiments on the assembly lines. Table 6-5 presents an overview of the experimental design including the experimental phases and experiments per phase.

Table 6-5. Overview of Experimental Design

Experimental Phase	Experiment
Setup Experiments	1. Choosing the Solution Approaches Parameters 2. Analysis of the Neighbourhood Solutions 3. Evaluation of the Solution Approach Configurations
Model Alternative Experiments	4. Simplified Model with Inventory Exclusion 5. Scalability of Best Solution Approach Configuration 6. Evaluation of Best Solution Approach Configuration in Real-Life Case

The first experimental phase initialises the parameters of the solution approaches, analyses the logic behind the neighbourhood operators and identifies the best solution approaches. The phase has 3 experiments. Experiment 1 focuses on choosing the optimal parameters for each solution approach. Selecting appropriate parameter for each assembly line is crucial for generating the best possible schedules while considering the computational time.

Experiment 2 analyses the logic behind the selection of better neighbourhood solutions. Our objective is to minimize inefficient moves and swaps. This allows us to evaluate a greater number of potentially superior schedules within the same amount of time. We observe the effect of excluding inefficient neighbourhood solutions. Therefore, we use similar solution approach parameters.

Experiment 3 focuses on the evaluation of the solution approach configurations to create the best possible schedule. Each configuration includes a constructive heuristic, improvement heuristic, and neighbourhood structure. We aim to identify the best solution approach per assembly line. Additionally, we analyse the individual effect of the constructive heuristics, improvement heuristics, and neighbourhood structures. Section 6.2.1 elaborates upon the specifics of Experiment 1, 2 and 3.

The second experimental phase considers model alternatives. Experiment 4 introduces a model alternative that excludes the inventory levels of the model during the construction phase of the solution approach configuration. The purpose of excluding the inventory levels is to reduce computational time. The experiment aims to evaluate the trade-off between the reduction in computation time and the potential loss in schedule quality. We apply the standard model once the construction of the schedule is completed. We make a comparison between the schedules created with the standard model and the simplified model. Additionally, we use the standard model in the improvement phase of the solution approach to see whether it recovers from the potential quality loss.

Experiment 5 evaluates the scalability of the solution approaches. This experiment assesses the performance and efficiency of the solution approaches to create a monthly schedule. The monthly scheduling period is 4.5 times longer than the weekly scheduling period. We update the problem instances for this experiment. We apply the best solution approach configurations from Experiment 3 to the monthly instances. Additionally, we analyse whether we update the solution approach parameters create higher quality schedules. Again, we make a trade-off between the quality of the schedule and the computational time.

Experiment 6 compares an actual schedule created by the planner with a schedule created using the best solution approach. The aim is to determine whether the best solution approach improves the schedules created by the planner. Additionally, this experiment shows if scheduling on a more extended time frame yields potential benefits. Section 6.2.2 explains Experiment 4, 5 and 6 in more detail.

6.2.1 Experimental Phase 1: Setup Experiments

This section explains the setup experiments in more detail. A proper setup of the experiments is important to create the best schedules for each assembly line. In this phase we initialise the parameters of the solution approaches, analyse the logical neighbourhood operators and identify the best solution approaches.

Experiment 1: Choosing the Solution Approach Parameters

In Experiment 1, we choose the solution approach parameters. We have a total of 3 solution approaches with parameters. The adjusted NEH with position exclusion (adjNEH), the Simple Improvement heuristic (SIH) and Simulated Annealing (SA). We determine the parameters for each solution approach per assembly line. We make a trade-off between the quality of the schedule and the running time. We use a set of 5 problem instances per assembly line to determine the parameters per solution approach. We use AS2 as an example in the process of selecting the parameters of the 3 solution approaches.

Adjusted NEH with Position Exclusion

The adjNEH constructive heuristic involves 1 parameter, the evaluation radius. The evaluation radius determines the range of preceding and consecutive positions that we evaluate next to the expected positions on a workstation. The other scheduling positions are excluded. We exclude positions to save running time. We make a trade-off between the running time and the quality of the schedule. We use the makespan, the most important objective, to determine the quality of the schedule. Figure 6-1 presents average makespan and running time when we apply different evaluation radii. The makespan stops decreasing after an evaluation radius of 1. The running time increases till an evaluation radius of 3. With both an evaluation radius of 3 and 4 we evaluate all possible positions. These settings show similarities with the NEH, constructive heuristic that also evaluates all possible scheduling positions. We select an evaluation radius of 1 for the adjNEH constructive heuristic.

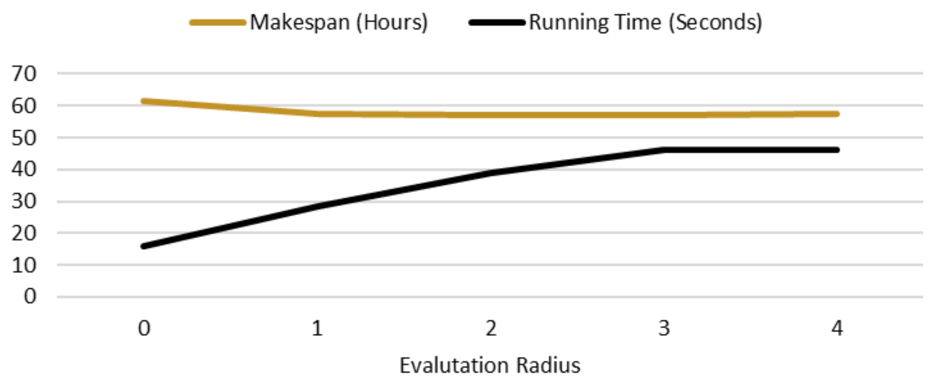


Figure 6-1. Evaluation Radius Analysis of the Adjusted NEH

We also perform this experiment for the other assembly line. Table 6-6 shows the objectives of the adjNEH with the selected parameters per assembly line. Additionally, it shows the objectives when we use the NEH constructive heuristic.

Table 6-6. Adjusted NEH Parameter

Assembly Line	Solution Approach	Evaluation Radius	C_{max} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}	T_{run} (Seconds)
AS1	NEH	All	45.00	0.90	19.20	1.10	1028.88
	adjNEH	0	45.00	0.90	19.20	1.10	321.59
AS2	NEH	All	57.29	1.20	9.40	1.20	44.15
	adjNEH	1	57.65	1.20	8.80	1.20	26.35
AS3	NEH	All	56.58	5.00	26.17	9.17	5042.97
	adjNEH	7	61.59	5.33	5.33	23.00	2274.66

We conclude the following from this experiment:

- The evaluation radius is 0 of AS1. We schedule a job always on an expected position in a workstation. There is no benefit in evaluating additional positions beyond the expected positions since each stage has the same sequence of jobs and each assembly time is the same due to the cycle time. We do not have to evaluate other positions.
- AS2 has an evaluation radius of 1. Each NEH iteration we evaluate the expected position and the positions before and after the expected position. We reduce the running time with roughly 40% with a quality loss of less than 1%.
- AS3 has an evaluation radius of 7. Even with this evaluation radius the adjNEH is not able to create a schedule with similar objectives to the NEH heuristic.
- We see a correlation between the size of the assembly line and the evaluation radius. The more jobs to schedule, the higher the evaluation radius. Additionally, the computational time increases when the number of jobs increases.

Simple Improvement Heuristic

The SIH has 1 parameter, the running time. We conduct an analysis to determine the running time of the SIH. Again, we compare the running time with the quality of the schedule. We select a running time which further increases in running time do not lead to substantial improvement in the quality of the schedule. We use the LS constructive heuristic to construct a schedule as an input for the SIH. We use LS since it generates a relatively good schedule within a relatively short time. Figure 6-2 shows the average makespan, our main objective, over the time.

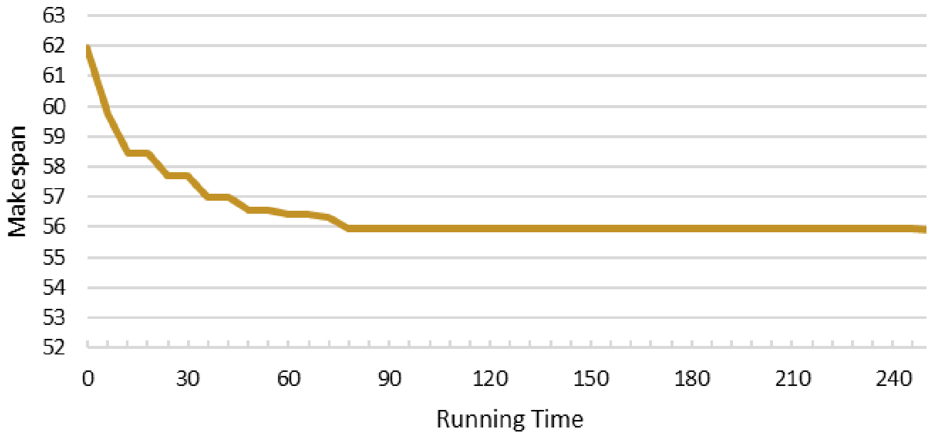


Figure 6-2. Converging Makespan with SIH

We see that the objective keeps improving in the first 75 seconds. Therefore, we select a running time of 90 seconds for the SIH of AS2. Table 6-7 presents the selected running times per assembly line.

Table 6-7. SIH Parameter

Assembly Line	T_{run} (seconds)
AS1	150
AS2	90
AS3	600

We conclude the following from this experiment:

- The SIH has a running time of 150 seconds for AS1. We cannot improve the makespan since each assembly time equals the cycle time. The algorithm improves the number of jobs that are on time and completed without shortages.
- The running time of the SIH equals 90 seconds for AS2. After 90 seconds there is no improvement.
- The SIH has a running time of 600 seconds for the AS3.
- There is a correlation between size of the assembly line instance and running time. The more jobs to schedule, the higher the running time.

Simulated Annealing

The SA improvement heuristic has 4 parameters: the start temperature T_{start}, end temperature T_{end}, the Markov chain length M, and the decrease factor α. Section 5.2.3 explains SA and the influence of the parameters on the process of exploration, exploitation and running time. We make a trade-off between the running time and the quality of the schedule. Again, we use AS1 as an example of the selection of the parameters. First, we calculate the acceptance ratios of different temperatures. Figure 6-3 presents the neighbourhood acceptance ratio of different temperatures.

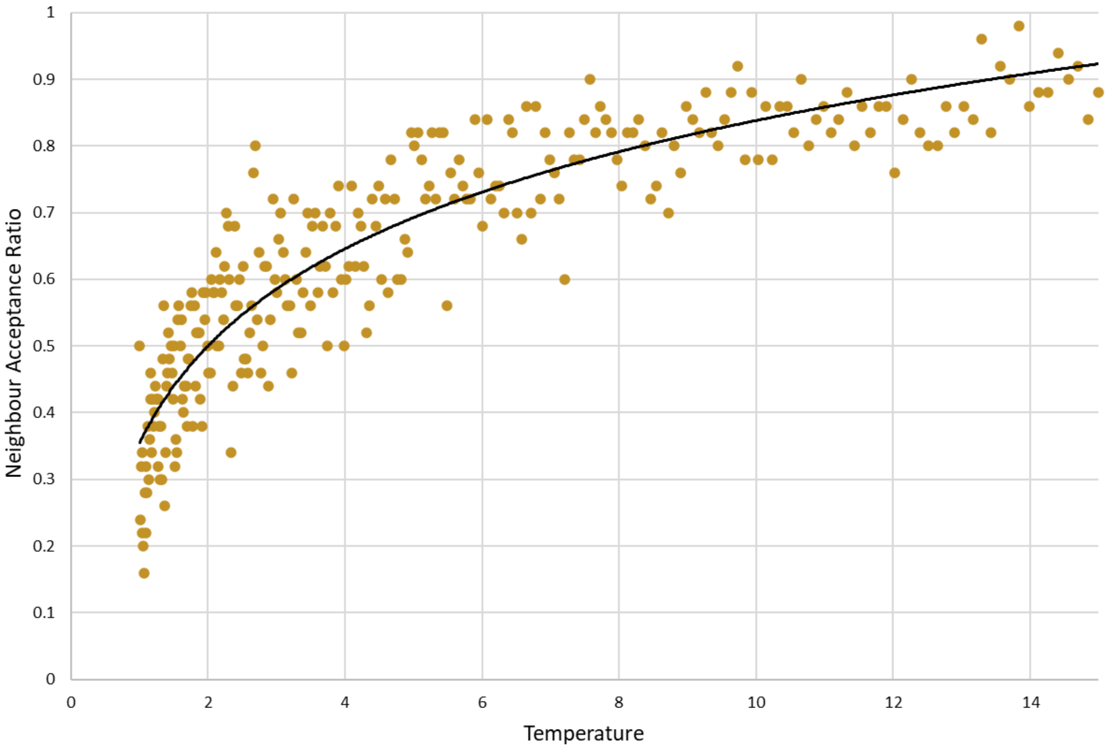


Figure 6-3. Neighbour Acceptance Ratio Per Temperature

The NEH, LS and adjNEH constructive heuristics construct schedules based on certain logic. Section 5.1 describes the logic behind these constructive heuristics. When we select a high T_{start} with a relatively good schedule there exist a high change of throwing away the good solution and explore a bad solution neighbourhood. This results in inefficient running time. Therefore, we do not want a high initial acceptance ratio. This reduces the change of accepting a much worse neighbouring solution. We continue experiment with different settings of the T_{start}, M and α. We select a T_{end} of 1. Table 6-8 presents the average makespan under different SA parameter settings for AS2.

Table 6-8. Makespan under Different SA Parameter Settings

T_{start}	M	α	C_{max}	T_{run}
2.5	50	0.6	61.33	23.71
2.5	50	0.9	59.32	44.70
2.5	50	0.95	58.16	70.34
2.5	75	0.6	59.59	37.72
2.5	75	0.9	58.03	102.04
2.5	75	0.95	57.30	180.03
4	50	0.6	59.08	43.62
4	50	0.9	58.21	129.14
4	50	0.95	56.99	230.23
4	75	0.6	57.31	100.66
4	75	0.9	55.63	341.88
4	75	0.95	55.03	671.41
10	50	0.6	58.24	51.29
10	50	0.9	56.76	175.06
10	50	0.95	55.33	321.02
10	75	0.6	56.88	118.04
10	75	0.9	55.25	491.99
10	75	0.95	55.07	919.07

We select the These parameters result in a reasonable objective within a reasonable running time. Table 6-9 presents the SA parameters of each assembly line.

Table 6-9. SA Parameters per Assembly Line

Assembly Line	T_{start}	T_{end}	M	α	T_{run}
AS1	4	1	50	0.8	594.57
AS2	4	1	75	0.9	341.88
AS3	4	1	50	0.8	1044.34

We conclude the following from these results:

- We cannot improve the makespan of AS1 due to the fixed cycle time. There is minimal improvement of the other objectives. Therefore, the SA parameters are relatively low for the AS1.
- The problem size also influences the running time. The running time of a larger problem size under the same SA parameters is much longer. This is visible in the selection of parameters for AS2 and AS3. AS3 has higher parameter. However, the running time is lower.

Experiment 2: Analysis of the Neighbourhood Solutions

This experiment analyses the selection of neighbourhood solutions to eliminate inefficient neighbourhood solutions. Section 5.2.1 introduces 4 neighbourhood operators to create the neighbourhood structure. The first neighbourhood operator swaps the operations of two jobs in each stage. The second neighbourhood operator only swaps the operations of 2 jobs in a single stage. The third neighbourhood operator moves the operation of a job in each stage to a different position. The fourth neighbourhood operator moves an operation of a job in a single stage to a different position.

Experiment 2 analyses the pattern behind the usage of these operators and which neighbourhood solutions result in better schedules. This helps us to identify which neighbourhood solutions we exclude when we select a new solution from the neighbourhood structure. This allows us to evaluate a greater number of potentially superior schedules within the same amount of time. This is particularly beneficial for larger size problems where many neighbourhood solutions are available. We use a set of 5 instances per assembly line to evaluate the patterns behind the neighbourhood operators.

Table 6-10 presents the analysis of the selection of neighbourhood solutions. It shows per operator which jobs are selected to generate a better neighbourhood solution. It also shows the positions where these jobs are in. Lastly, it presents the acceptance percentage per operator. This percentage represents the frequency with which this operator leads to an improved neighbourhood solution.

Table 6-10. Analysis Neighbourhood Structure

Operator	Job(s) Selected	Position(s) Selected	Operator Acceptance
Full Swap	○ One job with one job	○ All positions	AS1: 100% AS2: 44.71% AS3: 42.56%
Single Swap	○ One job with one job	○ Swap with jobs in preceding and consecutive positions on the same workstation. ○ Swap with jobs in positions with the nearest same start time on different workstations.	AS1: 0% AS2: 32.94% AS3: 51.54%
Full Move	○ Jobs with an operation on the fullest workstation in any of the stages	○ Move a job to any position in each workstation except from a position on the fullest workstation. The position in the other stages is determined by synchronizing the start and end times of successive stages.	AS1: 0% AS2: 7.06% AS3: 0%
Single Move	○ Jobs with an operation on the fullest workstation in any of the stages	○ Move to a position on different workstations with the start time closest to the end time of the job in the preceding stage.	AS1: 0% AS2: 15.29% AS3: 5.90%

We conclude the following from these results:

- ○ There is no specific pattern in the selection of jobs or positions for full swap neighbourhood operator. The swap of all operations of 2 jobs results potentially in a better schedule. This neighbourhood operator is always used for the AS1. This is because AS1 follows the same sequence in each stage and workstation. It also has a relatively high acceptance for AS2 and AS3.
- ○ A swap within a single stage happens only with jobs that are adjacent to each other on a workstation or have similar start times at different workstations within the same stage. Both

AS2 and AS3 select this operator a significant amount of time. However, AS3 tends to use this operator more often due to its greater number of workstations, providing more options for job swapping.

- The full move neighbourhood operator moves a job from the fullest workstations per stage to a position on a different workstation. Additionally, we move this job also in the other stages. In the preceding stages we select the position where we the end time aligns closely with the start time of the successive stage. In the successive stages we select the position where the start time aligns closely with the end time of the preceding stage. This maintains the continues flow of the job throughout the assembly line. Only AS2 utilizes the full move operator, as AS2 has a relatively equal and low number of stages and workstations. The overall flow of the assembly line is not significantly disrupted by performing moves in all stages.
- The single move neighbourhood operator moves a job from the fullest workstation to a different workstation in a single stage. It selects the position on a different workstation with the closest start time to the end time of the same job in the preceding stage. Both AS2 and AS3 use this neighbourhood to create better neighbourhood schedules. However, it is chosen less often than the swap neighbourhood operators.

We use these neighbourhood operators to create obvious neighbourhood solutions (ONS). In Experiment 3 we evaluate the effect of only considering these ONS. We compare ONS with the entire neighbourhood structure (ENS).

Experiment 3: Evaluation of the Best Solution Approach Configurations

This experiment evaluates the performance of the alternative model configurations introduced in Chapter 5. An alternative solution approach configuration consists of a constructive heuristic, improvement heuristic and a neighbourhood structure. We consider 4 constructive heuristics: the random, NEH, LS and the adjNEH constructive heuristic. We consider the SIH and SA as the 2 improvement heuristics. Additionally, we use 2 neighbourhood structures. We use the entire neighbourhood structure (ENS) and obvious neighbourhood structure (ONS) to generate neighbourhood solutions.

We compare and analyse the outcomes to determine the best solution approaches in terms of the objective values. We make a trade-off between the quality of the schedule and the computational time. We express the quality of the schedule in the objective values. We use a new set of 10 problem instances per assembly line. These problem instances differ from the problem instances of Experiment 1 and 2. We present the 5 most important objectives and the running time per solution approach configuration since these are relevant for the conclusions.

AS1

This section presents the results of the experiment of the solution approach configurations for the weekly schedule of AS1.

Table 6-11 presents objective values per solution approach configuration.

Table 6-11. Results Solution Approach Configurations Experiment AS1

Solution Approach Configuration	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}	T_{run} (Seconds)
RCH + None	178.90	138.59	0.00	0.00	1.20	1.98
NEH + None	45.00	15.00	0.90	19.20	1.10	1028.88
LS + None	45.00	15.00	0.90	22.00	1.30	90.08
adjNEH + None	45.00	15.00	0.90	19.30	1.10	321.59

RCH + SIH + ENS	144.80	109.51	0.00	0.00	1.60	152.83
NEH + SIH + ENS	45.00	15.00	1.70	20.50	1.80	1179.70
LS + SIH + ENS	45.00	15.00	1.50	22.30	1.60	240.67
adjNEH + SIH + ENS	45.00	15.00	1.50	20.00	1.60	472.97
RCH + SIH + ONS	178.65	138.46	0.00	0.00	1.80	153.12
NEH + SIH + ONS	45.00	15.00	2.00	21.70	2.00	1179.39
LS + SIH + ONS	45.00	15.00	2.10	22.40	2.10	240.89
adjNEH + SIH + ONS	45.00	15.00	1.80	21.80	1.80	472.23
RCH + SA + ENS	143.60	110.33	0.00	0.00	1.30	266.44
NEH + SA + ENS	45.00	15.00	2.10	21.50	1.80	2313.58
LS + SA + ENS	45.00	15.00	2.10	22.40	1.70	513.19
adjNEH + SA + ENS	45.00	15.00	1.90	21.80	1.60	682.52
RCH + SA + ONS	178.50	138.69	0.00	0.00	1.70	2333.41
NEH + SA + ONS	45.00	15.00	1.70	22.30	2.10	513.17
LS + SA + ONS	45.00	15.00	2.00	22.20	2.00	684.57
adjNEH + SA + ONS	45.00	15.00	1.80	22.00	1.80	2333.41

We observe the following from the results:

- The total makespan and average makespan equal 45 and 15 hours respectively. This is because the assembly line has a cycle time of 2.5 hours, and each job moves to the next stage simultaneously. The sequence-restricted jobs have zero to minimal influence on the schedule. The number of jobs that cannot be placed rightly after each other is low. There exists always a feasible sequence of jobs since the makespan equals 45 hours. This is not the case when we generate a schedule with the RCH.
- The usage of the ONS performs minimally better than the ENS. The ONS only allows full swaps for AS1. The ENS performs better when we use the RCH. In this case, we also need the other operators to improve the schedule. Therefore, the effectiveness of the ONS is more evident when it is employed alongside a constructive heuristic that creates a reasonable schedule.
- The LS constructive heuristic outperforms all other constructive heuristics in terms of the objectives. Its primary objective is to arrange jobs with near due dates and a high percentage of available components toward the start of the schedule and vice versa. Additionally, the running time is significantly lower than the other constructive heuristics based on logic.
- We see that the improvement heuristics cannot find better solutions in terms of the makespan. The number of jobs on-time and/or without shortages increases. There is no difference between the objectives of the SIH and SA, despite the running time.
- The usage of the ONS perform slightly better than the ENS. This is the case for every combination of constructive and improvement heuristic.

The LS constructive heuristic in combination with the SIH and ONS is the best solution approach configuration to create the schedules of AS1. This solution approach configuration results in the best objective with a reasonable running time. We use this solution approach configuration in Experimental Phase 2.

AS2

This section presents the results of the experiment of the solution approach configurations for the weekly schedule of AS2. Table 6-12 presents the objective values per solution approach configuration.

Table 6-12. Results Solution Approach Configurations Experiment AS2

Solution Approach Configuration	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}	T_{run} (Seconds)
RCH + None	121.52	68.39	0.20	2.30	1.10	0.26
NEH + None	57.29	30.71	1.20	9.40	1.20	44.15
LS + None	62.91	30.35	1.00	10.30	1.10	16.64
adjNEH + None	57.65	31.08	1.20	8.80	1.20	26.35
RCH + SIH + ENS	73.49	33.45	1.60	8.20	1.80	90.36
NEH + SIH + ENS	56.06	29.99	1.10	10.10	1.10	134.28
LS + SIH + ENS	57.42	30.09	1.20	9.60	1.20	93.43
adjNEH + SIH + ENS	56.81	30.27	1.10	10.00	1.10	117.46
RCH + SIH + ONS	65.68	31.67	1.60	9.20	1.80	90.42
NEH + SIH + ONS	54.70	30.57	1.60	10.40	1.60	141.48
LS + SIH + ONS	56.74	29.59	1.20	9.80	1.20	93.39
adjNEH + SIH + ONS	55.23	29.94	1.20	9.80	1.20	116.43
RCH + SA + ENS	61.00	28.92	1.80	10.40	1.80	236.01
NEH + SA + ENS	54.50	30.35	1.60	10.40	1.60	286.58
LS + SA + ENS	56.43	29.58	1.80	10.20	1.80	247.46
adjNEH + SA + ENS	54.88	30.12	1.60	10.20	1.60	254.02
RCH + SA + ONS	59.89	30.28	1.60	10.00	1.60	234.38
NEH + SA + ONS	54.42	30.17	1.60	10.40	1.60	293.32
LS + SA + ONS	56.44	29.54	1.80	10.40	1.80	245.90
adjNEH + SA + ONS	55.28	30.06	1.60	10.00	1.60	256.15

We conclude the following from the results:

- The NEH constructive heuristic outperforms all other constructive heuristics. However, the adjNEH constructive heuristic result in almost identical objectives as the normal NEH constructive heuristic and it needs on average 40% less running time.
- The number of on-time jobs is higher with the LS constructive heuristic. Its primary objective is to arrange jobs with near due dates and a high percentage of available components toward the start of the schedule and vice versa.
- Both improvement heuristics can improve the schedule. Nevertheless, the improvement is minimal after the NEH, LS and adjNEH constructive heuristics. There is no difference between the objectives of SIH and SA, despite the running time.
- The usage of the ONS perform slightly better than the ENS. This is the case for every combination of constructive and improvement heuristic except from the RCH. The ONS only created neighbourhood solutions with full swaps. We need other neighbourhood solutions to recover from a schedule generated with the RCH.

The results show that the adjNEH+SIH+ONS solution approach configuration performs the best in terms of the objectives and the running time. We use this solution approach configuration in Experimental Phase 2.

AS3

This section presents the results of the experiment of the solution approach configurations for the schedule of AS3. Table 6-13 presents the objective values per solution approach configuration.

Table 6-13. Results Solution Approach Configurations Experiment AS3

Solution Approach Configuration	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}	T_{run} (Seconds)
RCH + None	187.41	155.17	0.00	0.00	27.83	1.14
NEH + None	56.58	31.03	5.00	26.17	9.17	5042.97
LS + None	57.53	28.02	7.67	36.00	9.00	798.81
adjNEH + None	61.59	33.45	5.33	23.00	9.00	2274.66
RCH + SIH + ENS	123.37	92.96	0.00	0.00	26.50	601.77
NEH + SIH + ENS	56.35	30.84	7.00	27.17	10.33	5643.51
LS + SIH + ENS	56.22	28.22	9.33	33.50	10.17	1399.31
adjNEH + SIH + ENS	59.54	32.75	7.00	25.33	10.17	2875.18
RCH + SIH + ONS	83.79	51.01	4.33	10.50	15.67	601.52
NEH + SIH + ONS	56.30	30.70	6.83	26.83	9.83	5643.76
LS + SIH + ONS	55.61	28.25	8.83	33.50	9.83	1399.39
adjNEH + SIH + ONS	58.68	32.47	6.50	25.83	9.00	2875.23
RCH + SA + ENS	124.57	90.63	0.00	0.00	28.33	1046.57
NEH + SA + ENS	56.57	30.54	6.67	27.34	10.20	6088.41
LS + SA + ENS	56.32	28.52	9.50	33.50	10.17	1844.28
adjNEH + SA + ENS	58.78	31.85	7.23	25.17	10.25	3320.03
RCH + SA + ONS	95.39	50.84	0.00	0.00	28.21	1046.63
NEH + SA + ONS	56.30	30.43	7.23	28.83	10.33	6088.38
LS + SA + ONS	56.24	28.61	9.50	33.20	9.83	1844.34
adjNEH + SA + ONS	58.06	32.37	7.38	25.33	9.13	3319.92

We observe the following from the results:

- The NEH constructive heuristic outperforms the other constructive heuristics in terms of the most important heuristic, the makespan. However, the difference with the LS constructive heuristic is minimal. Additionally, running time is extremely large.
- The number of on-time jobs without shortages is higher with the LS constructive heuristic than the NEH and adjNEH constructive heuristic. Its primary objective is to arrange jobs with near due dates and a high percentage of available components toward the start of the schedule and vice versa.
- There is minimal improvement possible with the improvement heuristics. There is no difference between the objectives of SIH and SA, despite the running time.
- The usage of the ONS is minimally better than the ENS. Only, when we start with the RCH. The ONS performs much better than the ENS.

The LS+SIH+ONS solution approach configuration performs the best. Additionally, this approach has the lowest running time. We use this solution approach configuration for the Experimental Phase 2.

6.2.2 Experimental Phase 2: Model Alternative Experiments

This section presents the second experimental phase, which focusses on exploring model alternatives for the scheduling problem. The first alternative focusses on model simplification. This simplification aims to reduce the computational time. The second alternative assess the scalability of the solution approach configurations. We evaluate whether the best solution approach configurations are suitable to create monthly schedules. Lastly, we compare the schedules generated by the best solution approach with a schedule created by the planners for validation. This shows the difference between the schedule generated by the planner and with our approach.

Experiment 4: Simplified Model with Inventory Exclusion

This experiment presents a simplified version of the model. The standard model updates the inventory levels over time. We keep track of the inventory levels in each iteration of the constructive heuristics to check whether parts are missing. Essential shortages result in the delay of the assembly of a job. The simplified model excludes the inventory levels and starts checking it after the complete construction of the schedule.

Section 1.2.1 presents that Terberg aims to generate a high-quality schedule in a reasonable amount of time. The simplified model does not keep track of the inventory levels over time in the construction of the schedule. We use the standard model to evaluate the constructed schedule with the simplified model. We evaluate both the reduction in computational time and the quality loss. Additionally, we use the standard model in the improvement phase of the solution approach to see whether it recovers from the potential quality loss. We make a trade-off between both aspects to determine whether the simplified version is applicable to our problem.

Table 6-14 shows the objective values and percentual differences per assembly line. The symbols (↑) and (↓) indicate a significant increase or decrease of the KPI with an alpha of 0.05. The absence of these symbols indicates no significant difference. The colours green and red represent a better and worse performance, respectively.

Table 6-14. Results Model Simplification Experiment

Assembly Line	**KPI**	**Constructive**			**Improvement**		
		Normal	Simplified	**Difference (%)**	Normal	Simplified	**Difference (%)**
AS1	C_{max} (Hours)	45	45	**0.00**	45	45	**0.00**
	J_{ot+ns}	1.50	1.10	**-26.67**	2.00	2.00	**0.00**
	T_{run} (Seconds)	630.37	304.78	↓ **-51.65**	780.72	455.29	↓ **-41.68**
AS2	C_{max} (Hours)	57.1	57.29	**0.33**	54.12	54.5	**0.70**
	J_{ot+ns}	1.20	1.20	**0.00**	1.20	1.20	**0.00**
	T_{run} (Seconds)	47.72	19.49	↓ **-59.16**	144.89	109.62	↓ **-24.34**
AS3	C_{max} (Hours)	62.93	64.58	**2.62**	60.88	61.28	**0.66**
	J_{ot+ns}	5.33	4.50	**-15.57**	8.17	8.17	**0.00**
	T_{run} (Seconds)	2240.66	1081.37	↓ **-51.74**	2841.37	1681.81	↓ **-40.81**

We conclude the following from the results:

- There are minimal differences visible between the objectives of the standard model and the simplified model after the constructive heuristic. These differences are not even significant. The average running time of the constructive heuristic decreases significantly. The decrease for the AS1, AS2 and AS3 is respectively 51.65%, 59.16% and 51.74% when we us the simplified version of the model.

- After the improvement heuristic the objectives of the normal and simplified model are almost identical. The computational time is decreased with 41.68%, 23.34% and 40.81% for the AS1, AS2 and AS3, respectively.

The results show that it is advisable to use the simplified model to construct schedules for each assembly line. This simplified approach notably reduces the computational time required for constructive heuristics. Furthermore, any slight difference in the objective values is mitigated by the subsequent application of the improvement heuristic. Both the LS and adjNEH constructive heuristics consider the shortages per job, the impact of the shortage and the supply date during the construction with Equation 5-3. The heuristic prioritise jobs based on the available of parts and consider a limited number of scheduling positions based on the prioritisation.

Experiment 5: Scalability of the Solution Approach Configurations

This experiment evaluates the scalability of the weekly solution approach configurations. Section 1.2.1 describes that Terberg wants to create a monthly schedule instead of a weekly schedule in the coming future. We are interested in the scalability of the solution approach configurations. Certain algorithms that perform well for smaller-scale problems might not scale equally effectively to handle larger problem instances. We evaluate the quality of the schedule to determine whether the solution approaches are scalable. Additionally, we consider the computational time to determine the scalability of the solution approaches. We update the parameters when the parameters for a weekly schedule cannot create a high-quality schedule. We use the same methods to update the parameters as explained in Experiment 1. Table 6-15 presents the initial parameter settings and the updated parameter settings per assembly line and solution approach. We do not change the selected neighbourhood structure.

Table 6-15. Updated Parameters per Assembly Line and Solution Approach

Assembly Line	Sol. Appr. Config.	Parameter	Initial Weekly Settings	Updated Monthly Settings
AS1	LS	-	-	-
	SIH	Running Time	150 seconds	675 seconds
AS2	adjNEH	Evaluation Radius	1 position	4 positions
	SIH	Running Time	90 seconds	405 seconds
AS3	LS	-	-	-
	SIH	Running Time	600 seconds	2700 seconds

For AS1 and AS3, we only update the SIH. Both assembly lines use the LS constructive heuristic to create an initial schedule. We update the running time of the SIH of AS1 to 675 seconds and of AS3 to 2700 seconds. Both the constructive and improvement heuristic of AS2 use parameters. We update the evaluation radius parameter of the adjNEH to 4 and the running time of SIH to 270 seconds.

We apply both the initial weekly parameters and the updated monthly parameters to the monthly instances of the 3 assembly lines. Table 6-16 presents the results of Experiment 4. The symbols (↑) and (↓) indicate a significant increase or decrease of the KPI with an alpha of 0.05. The absence of these symbols indicates no significant difference. The colours green and red represent a better and worse performance, respectively.

Table 6-16. Results Solution Approach Configuration Scalability Experiment

	KPI	Constructive	Improvement

Assembly Line		Weekly	Monthly	**Difference (%)**	Weekly	Monthly	**Difference (%)**
AS1	C_{max} (Hours)	160	160	**0.00**	160	160	**0.00**
	J_{ot+ns}	13.22	13.22	**0.00**	14.89	21.33	↑ **43.25**
	J_{ot}	117.89	117.89	**0.00**	116.89	118.67	**1.52**
	J_{ns}	13.22	13.22	**0.00**	14.89	21.56	↑ **44.80**
	T_{run} (Seconds)	3727.23	3727.23	**0.00**	3877.52	4552.61	↑ **17.43**
AS2	C_{max} (Hours)	190.64	187.15	**-1.83**	186.38	177.13	**-4.96**
	J_{ot+ns}	5.7	6.2	**8.77**	6.8	7	**2.94**
	J_{ot}	33.7	36.6	**8.86**	37.1	39.2	**5.66**
	J_{ns}	9	8.9	**-1.11**	9.2	8.9	**-3.26**
	T_{run} (Seconds)	160.39	384.02	↑ **139.43**	250.88	784.35	↑ **212.64**
AS3	C_{max} (Hours)	166.68	166.68	**0.00**	165.81	165.06	**-0.45**
	J_{ot+ns}	44.01	44.01	**0.00**	44	43.65	**-0.80**
	J_{ot}	148.00	148.00	**0.00**	144.6	140.70	**-2.70**
	J_{ns}	173.60	173.60	**0.00**	170	165.70	**-2.53**
	T_{run} (Seconds)	7047.81	7047.81	**0.00**	7648.03	9747.95	↑ **277.38**

We conclude the following from the experiment:

- The total makespan and average makespan of AS1 remains also equal in the monthly situation. The updated parameters for the SIH increase the number of on-time and no-shortage jobs with 43.25%.
- The total makespan of AS2 decreases with 4.96%. However, the total running time is 212.64% higher in comparison with the weekly settings of AS2. The number of on-time and no-shortage jobs does not increase significantly.
- There is no significant difference between all the objective values of AS3 when we use the weekly settings or the updated monthly settings.

The results show that updating the running time parameter of the SIH for AS1 yields significant benefits. The additional computational time results in an improvement for the schedule of AS1. Adjusting the parameters of AS2 and AS3 does not result in significant improvements. The problem instance may become too large to effectively improve the schedule. The jobs at AS2 and AS3 can have different positions in different stage. When we increase the number of jobs the number of scheduling positions becomes too large, and the solution approach is not able to optimize it in a reasonable amount of time.

Experiment 6: Evaluation of the Best Solution Approach in a Simplified Real-life Case

Experiment 6 focusses on the validity of the schedules produced through our best solution approach configuration. To evaluate this validity, we apply the best solution approach configuration derived from Experiment 3 to a simplified real-life case. In this evaluation, we employ the best solution approach to generate a schedule and subsequently compare it with the schedule created by the planner.

This comparison is conducted over a span of 3 consecutive weeks per assembly. We generate a schedule for the jobs for each week with the best solution approach configuration. Additionally, we also construct a schedule for the 3 consecutive weeks combined. We use the best solution approach configuration of Experiment 5 to generate this schedule. This comparison helps us to assess whether a scheduling over an extended time frame proves to be beneficial.

AS1

This section presents the results of the simplified real-life case of AS1. Experiment 3 shows that we use the LS+SIH+ONS solution configuration to create schedules for AS1. We use 150 seconds for the SIH for the weekly schedule and 675 seconds for the 3 weeks schedule resulting from Experiment 5. Table 6-17 presents the results of the real-life case experiment of AS1.

Table 6-17. Results Real-Life Case AS1

Schedule Period	**Scheduling Approach**	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}
Week 1	Planner	49.88	14.25	0.00	11.00	0.00
	Proposed	49.88	14.25	0.00	11.00	0.00
	Difference (%)	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Week 2	Planner	49.88	14.25	9.00	24.00	14.00
	Proposed	49.88	14.25	12.00	24.00	17.00
	Difference (%)	**0.00**	**0.00**	**42.86**	**0.00**	**21.43**
Week 3	Planner	49.88	14.25	20.00	27.00	24.00
	Proposed	49.88	14.25	23.00	27.00	28.00
	Difference (%)	**0.00**	**0.00**	**15.00**	**0.00**	**16.67**
All Weeks Individually	**Total Difference (%)**	**0.00**	**0.00**	**20.69**	**0.00**	**18.42**
All Weeks Combined	Planner	125.88	14.25	29.00	62.00	38.00
	Proposed	125.88	14.25	51.00	62.00	69.00
	Difference (%)	**0.00**	**0.00**	**75.86**	**0.00**	**76.32**

We conclude the following from the results:

- The total and average makespan cannot be improved since the assembly time of the jobs are fixed based on the assembly line.
- The number jobs of with no-shortages increases by the week since each week the missing items come in.
- The proposed scheduling approach configuration cannot create a better schedule for Week 1. In both Week 2 and 3 the number of on-time jobs with shortages increases with 3. This is mainly due to number of jobs with no-shortages increases by the proposed scheduling approach configuration since the number of on-time jobs does not increase.
- The total number of on-time jobs with no shortages increases with 6 when we optimize the weeks separately. The number of on-time jobs with no shortages increases with 22 jobs when we optimize all three weeks together. This demonstrates that there is exchange of jobs between the 3 weeks.

AS2

This section presents the results of the simplified real-life case of AS2. Experiment 3 presents that we use the adjNEH+SIH+ONS solution configuration to create schedules for AS2. We use an evaluation radius of 1 for the weekly schedules and 4 for the 3-week schedule. Additionally, we use a SIH running time of 60 seconds for the weekly schedule and 405 for the 3-week schedule. Table 6-18 presents the results of the real-life case experiment of AS2.

Table 6-18. Results Real-Life Case AS2

Schedule Period	Scheduling Approach	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}
Week 1	Planner	59.23	29.74	2.00	9.00	3.00
	Proposed	54.86	28.74	3.00	9.00	3.00
	Difference (%)	**-7.38**	**-3.36**	**50.00**	**0.00**	**0.00**
Week 2	Planner	57.08	25.87	1.00	10.00	1.00
	Proposed	51.75	26.21	1.00	10.00	1.00
	Difference (%)	**-9.34**	**1.31**	**0.00**	**0.00**	**0.00**
Week 3	Planner	52.08	25.30	1.00	7.00	2.00
	Proposed	48.00	25.14	2.00	7.00	2.00
	Difference (%)	**-7.83**	**-0.63**	**100.00**	**0.00**	**0.00**
All Weeks Individually	**Total Difference (%)**	**-8.18**	**-1.01**	**50.00**	**0.00**	**0.00**
All Weeks Combined	Planner	124.83	28.61	4.00	26.00	6.00
	Proposed	115.25	28.07	6.00	26.00	6.00
	Difference (%)	**-7.67**	**-1.89**	**50.00**	**0.00**	**0.00**

We conclude the following from the results:

- In each week, the proposed scheduling approach decreases the total makespan. For the single weeks it reduces the makespan with 14.26%, 9.34% and 7.84%. The makespan for the 3-week schedule reduces with 7.68%.
- The average makespan remains roughly equal for each week and the 3-week schedule.
- In both Week 1 and 3 the proposed scheduling approach reduces the number of on-time jobs with no shortages with 1. The number of on-time jobs with no shortages remains the same in Week 2.
- The number of on-time jobs with no shortages increases with 2 when we schedule over the 3-week period. This is the same number as when we create the schedule for individual weeks. There is minimal to no exchange between the individual weeks.

AS3

This section presents the results of the simplified real-life case of AS3. Experiment 3 presents that we propose the LS+SIH+ONS solution approach configuration to create schedules for AS3. We use a running time of 600 seconds for the SIH to create both the weekly schedule and the 3-week schedule. Table 6-19 presents the results of the real-life case experiment of AS3.

Table 6-19. Results Real-Life Case AS3

Schedule Period	Scheduling Approach	C_{max} (Hours)	C_{avr} (Hours)	J_{ot+ns}	J_{ot}	J_{ns}
Week 1	Planner	58.40	30.31	10.00	10.00	35.00
	Proposed	53.33	29.24	11.00	11.00	35.00

	Difference (%)	**-8.68**	**-3.53**	**10.00**	**10.00**	**0.00**
Week 2	Planner	56.17	31.86	4.00	4.00	31.00
	Proposed	52.22	31.17	6.00	6.00	31.00
	Difference (%)	**-7.03**	**-2.17**	**50.00**	**50.00**	**0.00**
Week 3	Planner	51.08	30.57	21.00	21.00	24.00
	Proposed	50.08	31.44	21.00	21.00	24.00
	Difference (%)	**-1.96**	**2.85**	**0.00**	**0.00**	**0.00**
All Weeks Individually	**Total Difference (%)**	**-6.05**	**-0.96**	**8.57**	**8.57**	**0.00**
All Weeks Combined	Planner	104.39	37.19	35.00	35.00	90.00
	Proposed	102.33	37.41	38.00	38.00	90.00
	Difference (%)	**-1.97**	**0.59**	**8.57**	**8.57**	**0.00**

We conclude the following from the results:

- In each week the proposed scheduling approach decreases the total makespan. For the single weeks it reduces the makespan with 8.68%, 7.02% and 1.96%. The makespan of the 3-week schedule reduces with 1.98%.
- The on-time jobs with no shortages only increases for Week 2. The number of on-time jobs with no shortages increases from 4 to 6 jobs. The number of on-time jobs with no shortage remain the same in the other weeks and in the 3-week schedule. There is no to minimal exchange of jobs between the individual weeks.

6.3 Chapter Conclusion

The goal of this chapter is to describe all relevant information to provide a clear answer to the following research question:

"Which alternative solutions approach performs best compared to each other and how does the best solutions approach perform under different experimental settings at Terberg?".

We perform experiments to answer this question. Section 6.1 explains the creation of the problem instances for the experiments. A problem instance exists of the assembly line, job, and part parameters. First, the assembly line parameters show per assembly line the number of stages, workstations, and jobs per job type. Additionally, it shows the assembly line-specific extensions. These extensions are the restricted workstations per stage, blocked consecutive jobs per stage, setup times per stage and drying time per stage. Second, the job parameters present per job the assembly time per stage and workstation, due date and the BOM. Third, the part parameters present per part the initial inventory, the replenishment moments, and the replenishment quantities.

Section 6.2 presents the experimental design and the results We divide the experiments in 2 phases, setup experiments and model alternative experiments. Both experimental phases contain 3 research questions. Additionally, we determine the lexicographical order of the objectives. The lexicographical order is $f_L(C_{max}, C_{avr}, J_{ot+ns}, J_{ot}, J_{ns}, H_{tl}, N_s, H_{ts})$.

Experiment 1 determines the variable parameters for the solution approaches that have variable parameters. We make a trade-off between the quality of the schedule and the computational time. Based on the experiments we conclude that simulated annealing is not a proper solution approach. The computation time of generating and evaluating a single schedule is too high to therefore not beneficial.

Experiment 2 analyses the underlying logic behind the neighbourhood operators to enhance computational efficiency. We conclude that specific moves and swaps consistently yield superior schedules. These patterns can very between different assembly line.

Experiment 3 evaluates the performance of the alternative solution approach configurations. An alternative solution approach configuration consists of a constructive heuristic, improvement heuristic and a neighbourhood structure. We make a trade-off between the quality of the schedule and the computational time. We conclude that LS+SIH+LO is the best solution approach configuration for AS1 and AS3. The adjNEH+SIH+LO solution approach configuration performs the best for AS2.

In Experimental Phase 2 we present model alternative experiments. Experiment 4 presents a simplified model that excludes the inventory levels of the model. In this model we do not keep track of the inventory levels over time. We conclude that we can exclude the tracking of the inventory levels over time since it reduces the computational time with 41.68%, 24.38% and 40.81% for AS1, AS2 and AS3, respectively. There is no without significant loss of quality after we perform the improvement heuristic.

Experiment 5 evaluates the scalability of the best solutions approach configurations from Experiment 3 to create a monthly schedule. We conclude that we need to update the variable parameters of the solution approaches to generate better monthly schedule for AS1. The scheduling period of AS2 and AS3 become too large when we create a monthly schedule.

Experiment 6 compares the schedule created by the planner with the schedule created by the best solution approaches configuration from Experiment 3 to validate them. We compare a 3-week schedule of the planner with the schedule created by the proposed approach per assembly line. We conclude that the proposed approach improves the real-life schedules. The jobs also exchange between weeks when we apply the proposed approach to the 3 weeks combined for AS1. This is not the case for the AS2 and AS3. There is no difference between the objectives when we apply the solution approach individually to the weeks or combined.

7 Conclusions & Recommendations

This chapter discusses the conclusions and recommendations derived from the research. Section 7.1 summarizes the main findings and answers the main research question. Section 7.2 describes the recommendations for Terberg Group B.V. Section 7.3 discusses the limitations of the research and elaborates upon future research topics. Lastly, Section 7.4 describes the contribution of this research to science.

7.1 Conclusion

The assembly subsidiaries of Royal Terberg Group have difficulties with creating a proactive assembly schedule. There is an absence of a scheduling approach that considers the variations, uncertainties and restrictions that are present at each assembly subsidiary. This results in inefficient work at the assembly lines, too many penalties due to late delivery and too much extra (re)scheduling work for the planner. Therefore, the research question is:

"What scheduling approach can be adopted by the assembly subsidiaries of Royal Terberg Group to effectively enhance the quality of the schedule, while considering the variations, uncertainties, and restrictions?

We express the quality of the schedule in KPIs. We use these KPIs to evaluate and compare schedules. The KPIs are optimized in lexicographical order, $f_L(C_{max}, C_{avr}, J_{ot+ns}, J_{ot}, J_{ns}, H_{tl}, N_s, H_{ts})$. We developed an encoding and decoding algorithm that efficiently determines the start and end times of jobs in the schedule based on a sequence of jobs on each workstation in each stage. This algorithm also tracks inventory levels throughout the scheduling period. We use the start- and end times and the inventory levels to calculate the KPIs of a generated schedule.

We perform experiments to select the best solution approach configuration per assembly line. A solution approach configuration consists of a constructive heuristic, an improvement heuristic, with a neighbourhood structure. We proposed a total of 20 solution approach configurations to create schedules for each assembly subsidiary. We consider the assembly line specific restrictions when we generate the schedules with these solution approaches. Table 7-1 presents the best solution approach configuration per assembly line.

Table 7-1. Summary of the Best Solution Approach Configuration per Assembly Line

Assembly Line	Constructive Heuristic	Improvement Heuristic	Neighbourhood Structure
AS1	LS	SIH	ONS
AS2	adjNEH	SIH	ONS
AS3	LS	SIH	ONS

We conclude that all assembly lines generate the best schedules with the obvious neighbourhood structure. The ONS exclude moves and swap that certainly do not generate better schedules. It only moves jobs to and swaps jobs in positions that potentially lead to improvement of the objective function.

Furthermore, the combination of the LS constructive heuristic and SIH yields the most effective weekly schedules for AS1 and AS3. For AS2, the NEH constructive heuristic in conjunction with SIH generates the best schedules. The difference between the best solution approach is mainly caused by the input and the restrictions of the assembly lines. At AS2, a high number of jobs are workstation restricted. Whereas a low number of jobs at AS1 and AS3 need specific workstations. The adjNEH constructive heuristics prioritises these workstation-restricted jobs by scheduling them first on the specific workstations. We select the solution approach configurations of Table 7-1 to assess the performance and feasibility under the following scenarios:

- Construct schedules with a simplified model that excludes the inventory levels;
- Determine the scalability of the solution approach configuration when applied to a monthly scheduling period;
- Compare the schedules with the schedule create by the planner in a real-life case.

The standard model updates the inventory levels over time. We keep track of the inventory levels in each in of the constructive heuristics to check whether parts are absent. Essential shortages delay the start time of an assembly. The simplified model excludes the inventory levels and starts checking it after the construction of the schedule.

We conclude that it is beneficial to exclude the inventory levels of the model during the construction of the schedule for all assembly lines. Both the LS and adjNEH constructive heuristics consider the shortages per job, the impact of the shortage and the supply date during the construction with Equation 5-3. The heuristic prioritise jobs based on the available of parts and consider a limited number of scheduling positions based on the prioritisation. The improvement heuristic can overcome the minimal significant difference that occurs due to the exclusion of the inventory levels of the problem. In the end, the exclusion reduces the total computational time with 41.68%, 24.34% and 40.81% for AS1, AS2 and AS3, respectively.

Terberg has a scheduling period of 1 week. In the future, Terberg has plans to increase the scheduling period to multiple weeks. Therefore, the scalability of the solution approach configuration is important. We increase the number of jobs from the weekly number of jobs to the monthly number of jobs. We update the parameters of the solution approaches when this results to a better monthly schedule.

We conclude that is beneficial to update the parameter of the best solution approach for AS1. On average, the number of on-time finished jobs with no shortages increases with 43.25%. There is no to minimal improvement in the objectives of AS2 and AS3 after we update the parameters. We conclude that the problem instance becomes too large too effectively improve schedule in a reasonable time. When we increase the instance size the number of schedule positions increases in each stage. Especially at the AS3, the problem instances are already relatively large for the weekly schedule. A situation occurs where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

Lastly, we put the proposed solution approach configurations into practical application within a real-life case. We compare the schedules created by the planner with the schedules created by the best solution approach per assembly line. We use the schedules of 3 consecutive weeks. For the 1-week scheduling periods, we compare the schedules of individual weeks with those generated with the proposed approach for the week. Additionally, for a 3-week scheduling period, we compare the combined schedules of these 3 weeks with the schedule created with the proposed approach for a month.

We conclude that the proposed solution approach configurations improve the quality of the schedule at each subsidiary. The KPIs improve compared to a schedule created by the planner in a real-life case of each subsidiary. However, there are differences between improvement when we apply the proposed model the single week or the 3 weeks combined. At AS1, the average number of on-time jobs with no shortages increases with 20.69% when the proposed scheduling approach is applied to 3 individual weeks. The number of on-time jobs with no shortage increases even more when we extend the scheduling period to cover all 3 weeks combined. When we schedule the 3-weeks combined, the number of on-time jobs with no-shortages increases with 75.86%. In the combined approach the jobs are not restricted to a specific earlier determined week and are scheduled in any of the 3 weeks. This observation leads to the conclusion that the initial job distribution over the weeks was not optimal. This implies that an extended scheduling period yields superior results in this case. The problem size of AS1 also allows scalability. The problem is relatively small since each stage has the same sequence of jobs and the jobs move after a pre-determined cycle time.

At AS2, we see an improvement of the makespan and the number of on-time jobs with no shortages. The improvement of these objective values is similar when we apply the proposed approach to the 3 weeks individually or combined. We conclude that the initial job distribution over the weeks was correct since there is no difference in improvement the proposed individual and combined approach. We conclude that in this case the proposed approach is scalable since the improvement is similar in both the individual and combined optimization. Nonetheless, the problem size becomes relatively large when aggregate the 3 weeks. This leads to a situation where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

At AS3, the total makespan decreases with 6.05% when we apply the proposed scheduling approach is applied to the 3 individual weeks. However, the makespan only decreases with 1.97% when we apply the proposed approach to the 3 weeks combined. The increasing number of on-time jobs with no shortages is equal when we apply the proposed approach individually to the 3 weeks or combined. We conclude that the initial job distribution over the weeks was correct since there is no difference in improvement the proposed individual and combined approach. We conclude that in this case the proposed approach is not scalable. The problem size becomes extremely large. This leads to a situation where the planner may struggle to maintain the comprehensive overview of the schedule due to the increased number of jobs in the schedule.

The experiments did not include AS4. However, we can still draw conclusion based on the conclusions of the other assembly line. AS4 shows resemblance with AS1. Both assembly lines have the same sequence of jobs in each stage and simultaneously move to the next stage after a specific cycle time. The key difference is the number of workstations per stage and AS4 has a stage with setup and drying times. Nevertheless, we conclude that same solution approach configuration of AS1 is applicable to AS4. This is the LS constructive heuristics and SIH with the ONS. We conclude that the ONS of AS1 is similar to the ONS of AS4 since the sequence of the jobs is similar in each stage.

All in all, the proposed solution approach generates schedules that consider the:

- Variation in due dates, assembly times and required parts;
- Uncertainty in availability of materials;
- Assembly line specific restrictions.

Additionally, we improve the (re)scheduling time. The developed model together with solution approaches significantly reduce the (re)scheduling time for the planner of each subsidiary.

7.2 Recommendations

This section presents the recommendations for Royal Terberg Group and the individual subsidiaries. We recommend implementing the scheduling approach that can optimize the quality of the schedule the most. This varies per assembly line. Therefore, this section provides recommendations per assembly line and general recommendations that are applicable to Royal Terberg Group.

We recommend the LS constructive heuristics and the SIH with the ONS for AS1. This solution approach configuration outperforms the other configurations and has a reasonable computational time. Additionally, this approach is also scalable to a longer scheduling period. It is advisable to increase the scheduling period to enable jobs to be flexible across multiple weeks when the initial distribution of jobs over the weeks is not optimal. The problem size of AS1 is scalable since the jobs are in the same sequence in every single stage.

Our recommendation for the schedule of AS2is to utilize the adjNEH constructive heuristic and the SIH with the ONS. This solution approach configuration yields the best objective value while maintaining a reasonable execution time. This is particularly advantageous considering that currently a high number of jobs require a specific workstation in at least 1 stage. We recommend that the planner of AS2 continues to maintain a strict maximum number of jobs per job type per week. This constraint is necessary to ensure the creation of an efficient schedule. By limiting the number of jobs per job type in each week, the planner can effectively manage the workload distribution and prevent overloading the job-restricted workstations. This approach is potentially scalable to a longer scheduling period. Nevertheless, an increase in the number of jobs can potentially lead to a loss of comprehensive oversight for the planner.

Our recommendation is to utilize the combination of the LS constructive heuristic and SIH with the ONS for AS3. This approach allows for the creation of appropriate schedules within a reasonable timeframe. If the number of workstation-restricted jobs increases, we potentially create better schedules with the NEH or adjNEH constructive heuristic. We also recommend investigating the workload division of AS3. We advise to decrease the assembly time by effectively assembling or decrease the number of jobs that have high assembly time in certain stages. Another option is to increase the number of workstations in these stages. The solution approach and the problem size are not scalable. The problem size of AS3 is already relatively large for the weekly schedule.

Our recommendation for scheduling AS4 is to employ the LS constructive heuristic in combination with the SIH using the ONS. This recommendation is grounded in the similarities shared between AS 1 and AS4. However, it is important to conduct further experiments once AS4 finalizes and verifies their assembly data.

Finally, we offer general recommendations for the Royal Terberg Group that are applicable to all subsidiaries within the scope of this research. These advises are intended to enhance the scheduling approach across all subsidiaries. Chapter 4 introduces that the problem considers the required parts and fluctuations of the inventory levels over time in the schedule. We advise to exclude the inventory levels in the construction of the schedule. This significantly saves computational time, and the quality loss is neglectable since the improvement heuristic mitigate the slight difference in the objective values.

The model relies on deterministic assembly times allocated to each job type. The actual assembly times may deviate from these values, depending on the parts that are used. We strongly recommend that each assembly subsidiary monitor the assembly times and establish clear correlations between these times and the required parts. This contributes to the generation of more accurate schedules.

7.3 Limitations & Future Research

This section highlights the limitations of the current research and suggests potential areas for future investigation to enhance assembly performance and improve the overall research.

First, we limit the research scope to the main stages of each assembly line. These stages are most critical in determining the overall makespan and have a significant impact on scheduling efficiency and performance. However, incorrect scheduling in one or multiple pre-assembly stages can lead to inefficiencies in the main assembly stages as well. Potentially, we can use backward scheduling to calculate maximal start times of the pre-assemblies. Future research is required whether this method is adequate or if the pre-assemblies need to be considered during the scheduling of the main assembly stages.

Second, the use of lexicographical order in the optimization imposes limitations on the research and restricts the exploration of trade-offs. Lexicographical order optimization involves prioritizing objectives or variables based on their order in a predefined list. This may not always reflect the complex relationships the objectives. The lexicographical order optimization may also overlook potentially valuable solutions that lie outside the predefined priority order. Future research needs to focus on determining the appropriate weights for each objective. The current lexicographical prioritization can serve as a starting point to determine the weights. A common approach is the Analytic Hierarchy Process (AHP). This method involves systematically assessing the relative importance of different objectives by relevant stakeholders and experts.

Third, we mentioned earlier that we use deterministic assembly times per job type. However, the assembly times can differ within a job type. We did not include this difference in assembly time. Future research should focus on better estimating the assembly times. This also incorporates better registration of assembly time by the assembly line workers.

Fourth, we implemented the adjNEH with position exclusion to reduce computational time. In this version of the adjNEH we only exclude positions horizontally. However, the adjNEH is still time consuming for extremely large instances, such as the schedule of AS3. Future research should focus on the exclusion of positions vertically when we construct a schedule. We exclude scheduling jobs on workstations that generate worse solutions. For example, we do not schedule a job in any position on the fullest workstation(s).

Lastly, we experiment with the logic behind operators. Our research only focusses on the positions the jobs are in and move to. We do not look at the aspects of jobs, e.g., due date, assembly times and/or availability of parts at that time. Further research could focus on integrating these job-specific data in the selection of potential moves and swaps of jobs. Certain positions in the schedule are excluded because they do not improve the schedule. Just as we did with the adjNEH constructive heuristic. This approach should save computational time.

7.4 Contribution of the Research

This section discusses the contribution of the research to the literature.

First, there is limited literature available about hybrid FSSPs that incorporate the usage of parts in the assembly process and keeping track of the inventory levels. Our problem deals with both product-specific and multi-purpose components. Therefore, our model keeps track of the inventory levels over time. Furthermore, our problem deals with different impact of shortages. When a component is unavailable, it leads to the delay of an assembly and potentially results in additional post-assembly time. Our model not only considers the assembly process delays but also strives to minimize the extent of extra post-assembly time.

Second, each assembly line within the scope of our problem has its own restrictions. The existing literature primary focuses on a single restriction per article. There is a gap in the literature where multiple restrictions are combined. Our model considers multiple restrictions. Additionally, we adjust the solution approaches to these restrictions. The NEH constructive heuristic considers the workstation-restricted jobs as the primary scheduling criteria. The solution approach prioritises the jobs with the least number of schedulable workstations a position in the schedule.

Third, our research introduces an adjusted version of the NEH constructive heuristic. This version does not yet exist in the literature. The adjNEH constructive heuristics shows reduces the computational time without significant loss of quality of the resulting schedule, depending on the problem size. Further research related to the exclusion of scheduling positions can reduce the computational time even more, considering the quality loss of the schedule.

Fourth, our research provides insights in the logic behind neighbourhood operators. It shows which moves and swaps of jobs in certain positions lead more likely to improve the schedule. Additionally, we indicate the selection percentage of these logic neighbourhood operators per assembly line. Certain operators are more beneficial for certain assembly line types.

Finally, this study contributes by demonstrating the effect of multiple constructive, improvement heuristics and neighbourhood structures on a real-life case study problem. The study reveals that different solution approaches are more suitable for specific types of assembly lines. The assembly lines have varying number of jobs to schedule and a different layout. Additionally, each assembly line has different restrictions.

Bibliography

Abdolrazzagh-Nezhad, M., & Abdullah, S. (2017). Job Shop Scheduling: Classification, Constraints and Objective Functions. *World Academy of Science, Engineering and Technology. International Journal of Computer and Information Engineering*, pp. 423-428.

Ahmadian, M. M., Khatami, M., Salehipour, A., & Cheng, T. C. (2021, March 24). Four decades of research on the open-shop scheduling problem to minimize the makespan. *European Journal of Operation Research 295*, pp. 399-426.

Al-Harkan, I. M., & Qamhan, A. A. (2019). Optimize unrelated parallel machines scheduling problems with multiple limited additional resources, sequence-dependent setup times and release date constraints. *IEEE Access*.

Al-Hinai, N. E. (2011, April 27). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, pp. 279-291.

B.V., P. M. (2022, May 9). Production and Planning at Terberg Benschop B.V. (R. Hampsink, Interviewer)

Becker, C., & Scholl, A. (2004). A survey on problems and methods in generalized asssembly line balancing. *European Journal of Operational Research*, 694-715.

Birgin, E. G., Ferreira, J., & Ronconi, D. P. (2014). *List scheduling and beam search methods for the flexible job shop scheduling problem with sequencing flexibility.* Sao Paulo: Department of Computer Science, Institute of Mathematics and Statistics, University of Sao Paulo.

Bowersox, D., & Closs, P. (1996). *Logistical management: The Integrated Supply Chain Process.* New York: McGraw-Hill.

Boysen, N., Fliedner, M., & Scholl, A. (2006). A classification of assemlby line balancing problems. *Production, Manufacturing and Logistics*, 674-673.

Brucker, P., Heitmann, S., & Hurink, J. (2003). Flow-shop problems with intermediate buffers. *OR Spectrum*, pp. 549-574.

Bultmann, M., Knust, S., & Waldherr, S. (2018, April 24). Flow shop scheduling with flexible processing times. *OR Spectrum*, pp. 809-829.

Can, K. C. (2008). *Postponement, Mass Custamization, Modularization and Customer Order Decoupling Point: Building the Model of Relationships.* Linköping University Department of Management and Engineering .

Chandra, P., Mehta, P., & Tirupati, D. (2004, July 06). *Permutation flowshop scheduling with earliness and tardiness penalties.* Indian Institute of Management Ahmedabad.

Deng, J., Wang, L., Wang, S.-y., & Zheng, X.-l. (2016). A competitive memetic algorithm for the distributed two-stage assemblyflow-shop schedulingproblem. *International Journal of Production Research*, pp. 3561-3577.

Engin, O., Ceran, G., & Yilmaz, M. K. (2011). An efficient genetic algorithm for hbyrid flow shop scheduling with mltiprocessor task problem. *Applied Soft Computering*, 3056-3065.

Gottlieb, j., Puchta, M., & Solnon, C. (2003). A study of greedy, local search, and ant colony optimization apporaches for car sequencing problems. *Applications of Evolutionary Computing*, 246-257.

Graham, R. (1991). Bounds on multiprocessing timing anomalies . *Information processing letters*, 291-297.

Gravel, M., Gagné, C., & Price, W. (2005). Review and comparison of three methods for the solution of car sequencing problem. *Journal of the Operational Research Society*, pp. 1287-1295.

Grodzevich, O., & Romanko, O. (2006). Normalization and other topics in multi-objective optimization.

Guzman, E., Andres, B., & Poler, R. (2021, December 22). Matheuristic Algorithm for Job-Shop Scheduling Problem Using a Disjunctive Mathematical Model. *Computers*.

Hamzadayi, A., & Yildiz, G. (2013). A simulated annealing algortihm based approach for balancing and sequencing of mxed-model U-lines. *Computers & Industrial Engineering*, pp. 1070-1084.

Han, W., Deng, Q., Gong, G., Zhang, L., & Luo, Q. (2021). Multi-objective evolutionary algorithms with heuristic decoding for hybrid flow shop scheduling problem with worker constraint. *Expert Systems with applications*.

Hans, E., Herroelen, W., Leus, R., & Wullink, G. (2007). A hierarchical approach to multi-project planning under uncertainty. *Internatial Journal of Management Science*, 563-577.

Hansen, P., Mladenovic, N., Todosijević, R., & Hanafi, S. (2016, August). Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*.

Hayes, A. (2020, November 8). *Investopedia*. Retrieved from Make to stock, Assemble to order, Make to Order .

Heerkens, H., & van Winden, A. (2017). *Solving Managerial Problems Systematically.* Groningen: Noordhoff Uitgevers.

Hosseinabadi, A. A., & Balas, V. E. (2016). A Novel Meta-Heuristic Combinatory Method for Solving Capacitated Vehicle Location-Routing Problem with Hard Time Windows. *Advances in Intelligent Systems and Computering*.

Jang, B., Burns, N., & Backhouse, C. (2004). Management of Uncertainty through Postponement. *International Journal of Production Research*, pp. 1049-1064.

Kis, T. (2003). On the complexity of car sequencing problm. *Operations Research Letters*, 13-17.

Komaki, G. M., Sheikh, S., & Malakooti, B. (2018, December 6). Flow shop scheduling problems with assembly operations: a review and new trends. *International Journal of Production Research*.

Komaki, G. S., & Malakooit, B. (2018). Flow shop scheduling problems with assembly operations: a review and new trends. *International Journal of Production Research*.

Laribi, I., Yalaoui, F., Belkaid, F., & Sari, Z. (2016, December). Heuristics for solving flow shop scheduling problem under resources constraints. *International Fedration of Automatic Control*, pp. 1478-1483.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computer & Industrial Engineering 37*, pp. 57-61.

Maravelias, C. T., & Sung, C. (2009, June 11). Integration of production planning and scheduling: Overview, challenges and opportunities. *Computers and Chemical Engineering*, pp. 1919-1930.

Naderi, B., Zandieh, M., & Roshanaei, V. (2009). Scheduling hybrid flowshops with seqeunce dependent setup times to minimize the makespan and maximum tardiness. *International Journal of Advanced Manufacturing Technonology*, 1186-1198.

Nagano, M., & Moccellin, J. (2002). A high quality solution constructive heuristic for flow shop sequencing. *The Journal of the Operational Research Society*, pp. 1374-1379.

Nawaz, M. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *International Journal of Management Science*, pp. 91-95.

Olhager, J. (2010, December). The role of the customer order decoupling point in production and supply chain management. *Computers in Industry*.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operational Research*, pp. 513-628.

Pang, X., Xue, H., Tseng, M.-L., Lim, M. K., & Liu, K. (2020). Hybrid flow shop scheduling problems using improved fireworks algorithm for permutation. *Applied Sciences*.

Rabbani, M., Ziaeifar, A., & Manavizadeh, N. (2014). *International Journal of Computer Integrated Manufacturing*, pp. 690-706.

Ravindran, D., Noorul Haq, A., Selvakuar, S., & Sivaraman, R. (2003, 16 May). Flow shop scheduling with multiple objective of minimzing makespan and total flow. *International Journal of Advanced Manufacturing Technology*, pp. 1007-1012.

Riane, F., Artiba, A., & Elmaghraby, S. E. (2010, November). Sequencing a hybrid two-stage flwoshop with dedicated machines. *International journal of production reserach*, pp. 4353-4380.

Rolf, B., Reggelin, T., Nahhas, A., Lang, S., & Müller, M. (2020). Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem. *Procedia Manufacturing*, pp. 442-449.

Ruiz, R., & Rodríguez, J. A. (2010). The hyprid flow shop scheduling problem. pp. 1-18.

Ruiz, R., Serifoglu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 1151-1175.

Ruiz, R., Serifoglu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 1151-1175.

Sadeh, N., & Fox, M. S. (1996). Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artifical Intelligence 86*, pp. 1-41.

Sarhadi, H., Chauhan, S. S., & Verma, M. (2020, March 21). A budget-constrained partial protection planning of a rail intermodal terminal network.

Schutten, J. (1996). List scheduling revisted . *Operations Research Letters*, 167-170.

Seker, S., Özgürler, M., & Tanyas, M. (2013). A weighted multiobjective optimization method for mixed-model assembly line problem. *Journal of applied mathematics*.

Shah-Hosseini, H. (2013, October). Multilevel Thresholding for Image Segmentation using the Galaxy-based Search Algorithm. *I.J. Intelligent Systems and Applications*, pp. 19-33.

Sivasankaran, P., & Shahabudeen, P. M. (2016). Heuristics for mixed model assembly line balancing problem with sequencing. *Scientific Research Publishing*, 41-65.

Stadtler, H., & Kilger, C. (2008). *Supply chain management and advanced planning : concepts, models, software, and case studies.* Berlin: Springer.

Talbi, E.-G. (2009). *Methaheuristics: From Design to Implementation.* Lille: John Wiley & Sons Inc.

Tang, L., Liu, W., & Liu, J. (2005). A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment. *Journal of Intelligent Manufacturing*, pp. 361-370.

Vank Hoek, R. (2000). The Thesis of Leagility Revisited. *International Journal of Agile Management Systems*, pp. 196-201.

Wang, K., Choi, S., & Qin, H. (2014). An estimation of distribution algorithm for hybrid flow shop shceudling under stocahstic processing times. *International Journal of Production Research*, pp. 7360-7376.

Wang, S., & Liu, M. (2012, July). A heuristic mehtod for two-stage hybrid flow shop with dedicated machines. *Computers & Operations Reserach*, pp. 438-450.

Yenisey, M. M., & Yagmahan, B. (2013). Multi-objective permuation flow shop scheduling problem: Literature review, classification and current trends. *Omega*, 119-135.

Yong, L., Zhantao, L., Xiang, L., & Chenfeng, P. (2022, September 2022). Heuristics for the Hybrid Flow Shop Scheduling Problem with Sequence-Dependent Setup times. *Mathematical Problems in Engineering*.

Zhang, G., Zhang, L., Song, X., Wang, Y., & Zhou, C. (2019). A variable neighborhood search based genetic algorithm for flexible. *Cluster Computing*, pp. 1561–1572.

Appendix A: Analysis of the Product Mix and Assembly Statistics

This appendix offers insights into the product mix and assembly times of each assembly line, which are integral to the research's scope. It provides a comprehensive overview of the product mix and assembly statistics. It shows the percentages of products assembled per product of 2022. It includes the average and standard deviation of the assembly times at each main- and substage of the assembly. Additionally, it presents the information regarding the number of product types that require each stage in the assembly process. This highlights the differences between the assembly volumes per product type and the corresponding assembly times and frequencies at both the main- and substages.

AS1

AS1 assemblies a total of 4 product types. Figure A-1 presents the percentage of the total production volume.

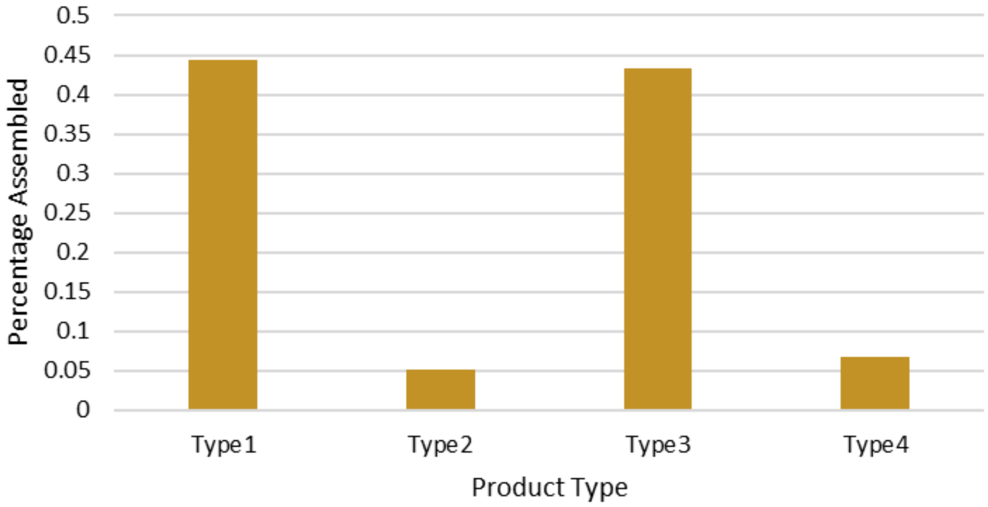


Figure A-1. Percentage per Product Type AS1

Table A-1 presents the assembly statistics of AS1. It shows the stage type, assembly times and number of product types that require the stage.

Table A-1. Assembly Stage Statistics AS1

Stage	Stage Type	Mean Assembly Time	Std. Assembly Time	Product Type Requiring Stage
Stage01	Main	7.08	0.00	4
Stage02	Main	7.08	0.00	4
Stage03	Main	7.08	0.00	4
Stage04	Main	7.08	0.00	4
Stage05	Main	5.23	0.51	4
Stage06	Main	7.08	0.00	4
Stage07	Sub	0.56	1.12	2
Stage08	Sub	1.75	0.00	1
Stage09	Sub	1.75	0.00	1
Stage10	Sub	2.66	1.65	2
Stage11	Sub	1.75	0.00	1
Stage12	Sub	2.38	0.12	4
Stage13	Sub	2.15	0.00	4

Stage14	Sub	5.69	0.00	4
Stage15	Sub	1.78	0.10	4
Stage16	Sub	5.66	0.00	4
Stage17	Sub	9.59	0.00	4
Stage18	Sub	0.55	1.60	2
Stage19	Sub	5.58	1.54	4
Stage20	Sub	1.39	0.38	4
Stage21	Sub	0.37	1.09	2
Stage22	Sub	0.21	0.61	2
Stage23	Sub	0.30	0.89	2

AS2

AS2 assemblies a total of 10 product types. Figure A-2 presents the percentage of the total production volume.

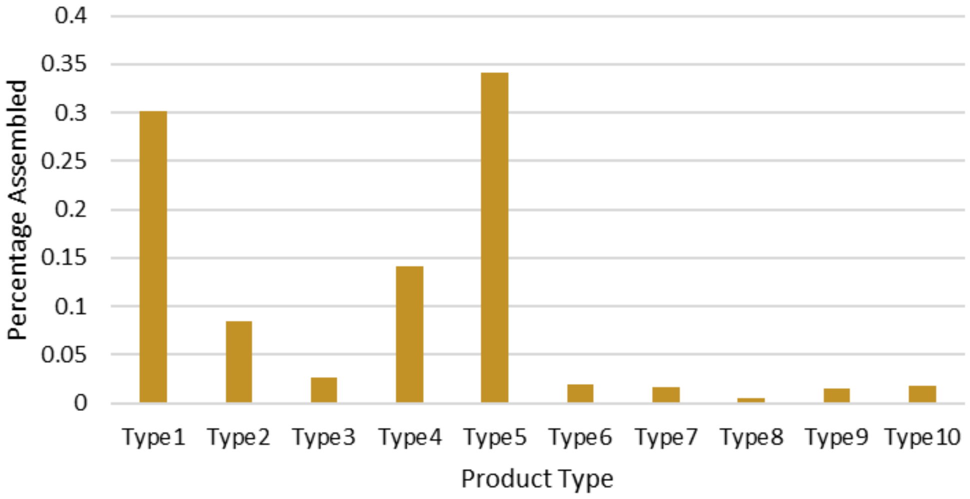


Figure A-2. Percentages per Product Type AS2

Table A-2 presents the assembly statistics of AS2. It shows the stage type, assembly times and number of product types that require the stage.

Table A-2. Assembly Stage Statistics AS2

Stage	Stage Type	Mean Assembly Time	Std. Assembly Time	Product Type Requiring Stage
Stage01	Main	19.28	6.91	10
Stage02	Main	24.86	5.14	10
Stage03	Main	17.41	4.64	10
Stage04	Sub	24.48	8.97	3
Stage05	Sub	3.23	0.54	10
Stage06	Sub	3.07	0.68	10
Stage07	Sub	4.35	1.45	10
Stage08	Sub	1.65	1.86	4
Stage09	Sub	9.37	3.35	10
Stage10	Sub	6.26	2.17	10
Stage11	Sub	15.79	5.01	10

Stage12	Sub	5.98	2.06	10
Stage13	Sub	4.61	2.47	8

AS3

Figure A-3 presents the percentage of the total production volume. Each type also has sub types.

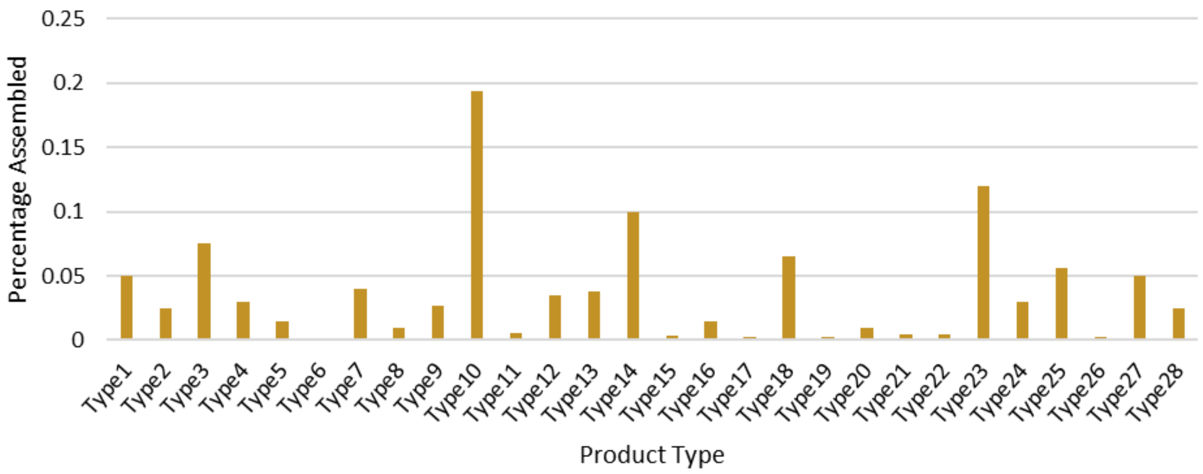


Figure A-3. Percentages per Product Type AS3

Table A-3 presents the assembly statistics of AS3. It shows the stage type, assembly times and number of product types that require the stage.

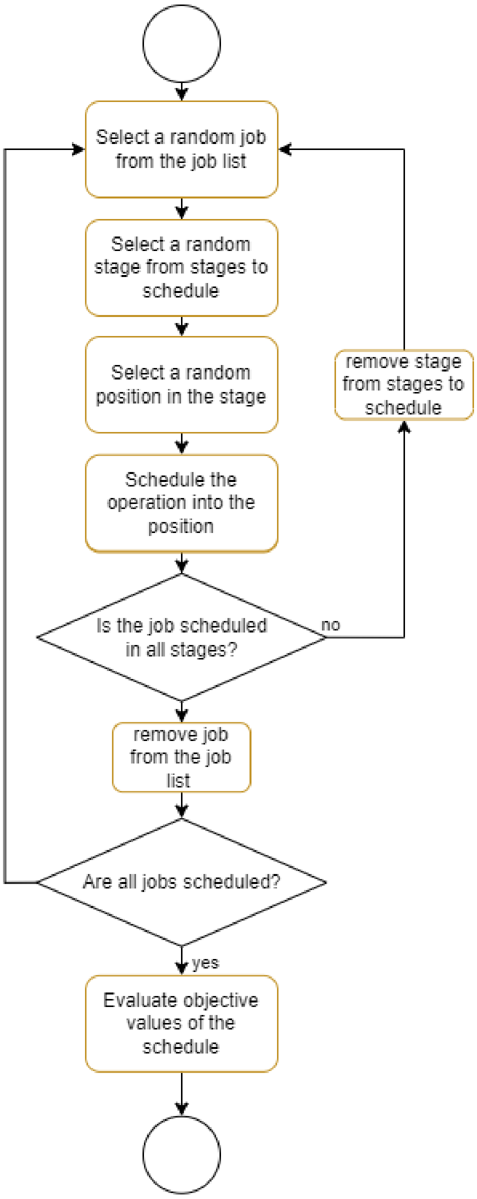
Table A-3. Assembly Stage Statistics AS3

Stage	Stage Type	Mean Assembly Time	Std. Assembly Time	Product Type Requiring Stage
Stage01	Main	4,21	1,37	55
Stage02	Main	3,67	0,75	55
Stage03	Main	7,56	1,99	55
Stage04	Main	5,04	1,53	55
Stage05	Main	5,37	1,46	55
Stage06	Main	1,05	0,21	55
Stage07	Sub	0,50	0,00	31
Stage08	Sub	1,69	1,90	31
Stage09	Sub	0,50	0,00	31

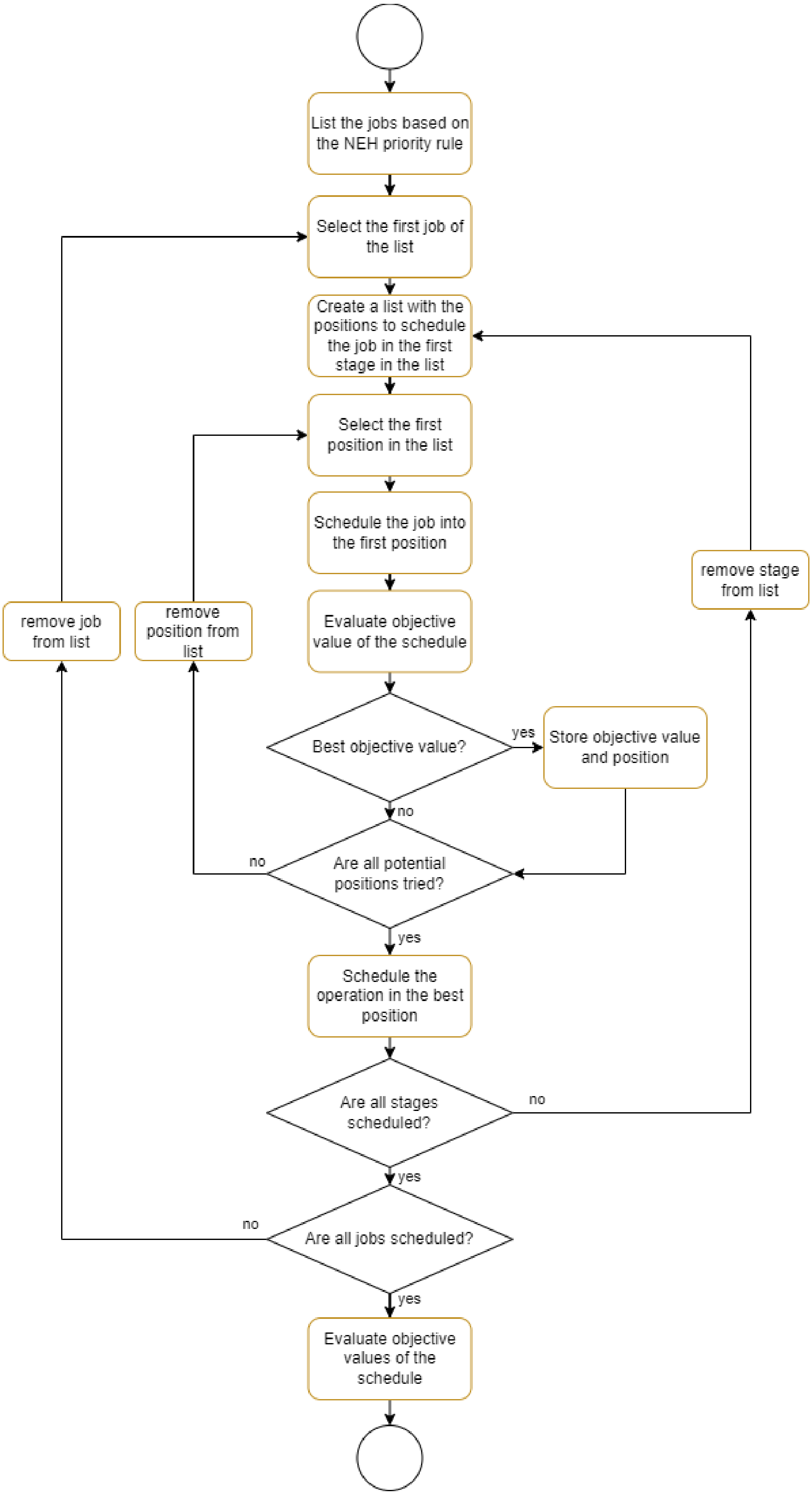
Appendix B: Flowcharts Approaches

This appendix presents the flowchart of each approach. There are 4 constructive and 2 improvement heuristics. The constructive heuristics are the random constructive heuristic, NEH heuristic, list scheduling heuristic and adjNEH heuristic.

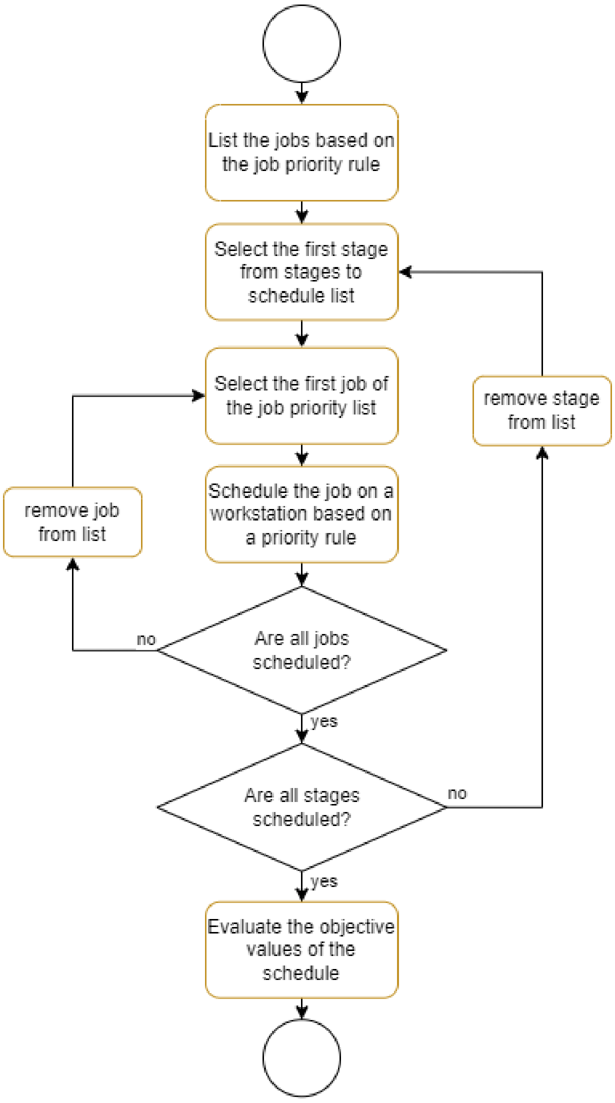
Random Constructive Heuristic


Select a random job from the job list
Select a random stage from stages to schedule
Select a random position in the stage
Schedule the operation into the position
Is the job scheduled in all stages?
no
remove stage from stages to schedule
remove job from the job list
Are all jobs scheduled?
yes
Evaluate objective values of the schedule

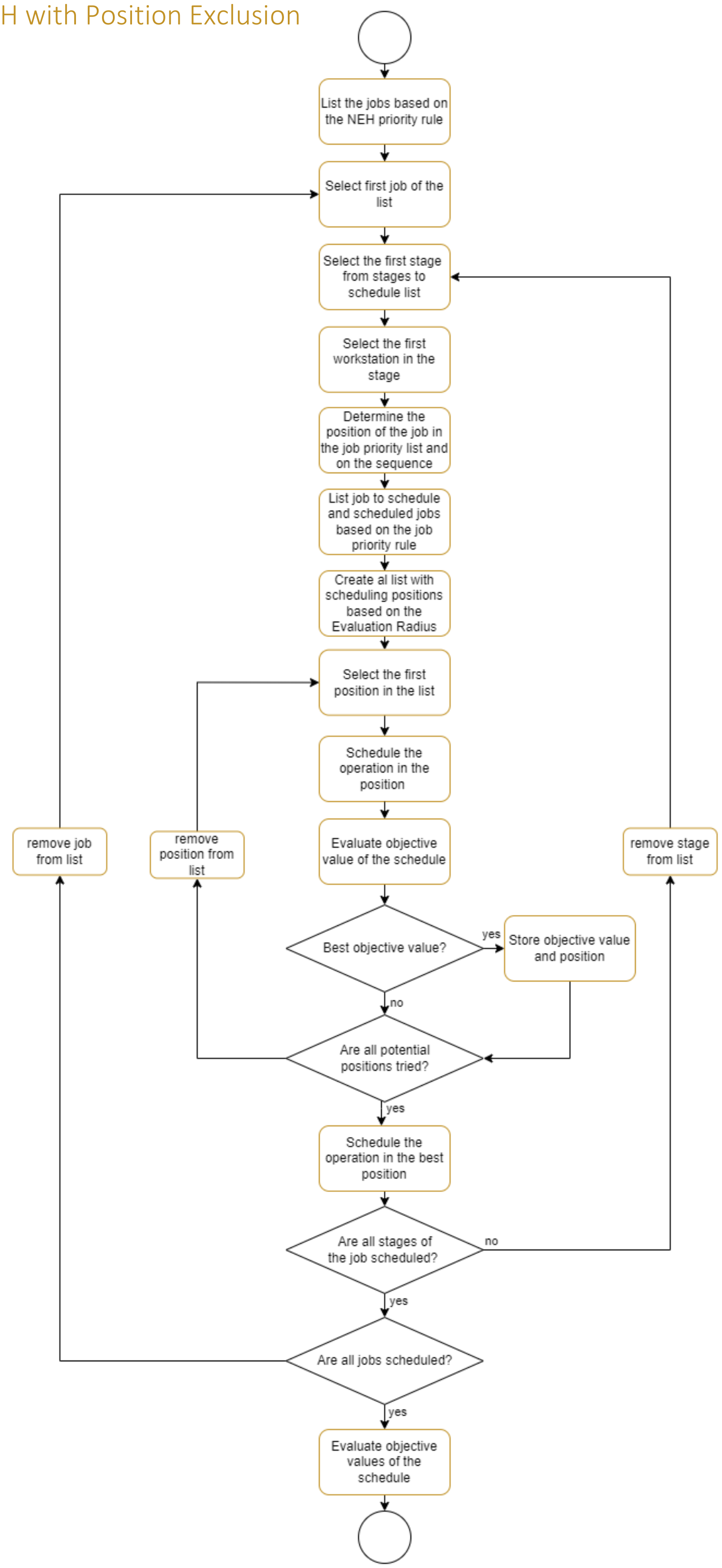
NEH Heuristic


List the jobs based on the NEH priority rule
Select the first job of the list
Create a list with the positions to schedule the job in the first stage in the list
Select the first position in the list
Schedule the job into the first position
Evaluate objective value of the schedule
Best objective value?
yes
Store objective value and position
no
Are all potential positions tried?
no
remove position from list
yes
Schedule the operation in the best position
Are all stages scheduled?
no
remove stage from list
yes
Are all jobs scheduled?
no
remove job from list
yes
Evaluate objective values of the schedule

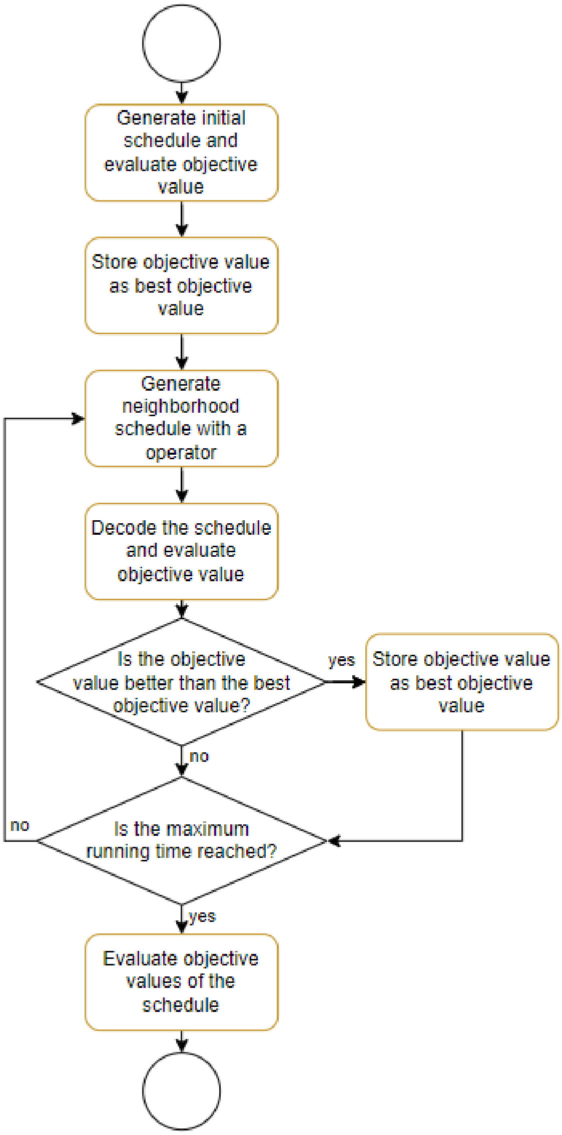
List Scheduling


List the jobs based on the job priority rule
Select the first stage from stages to schedule list
Select the first job of the job priority list
remove stage from list
remove job from list
Schedule the job on a workstation based on a priority rule
no
Are all jobs scheduled?
yes
Are all stages scheduled?
no
yes
Evaluate the objective values of the schedule

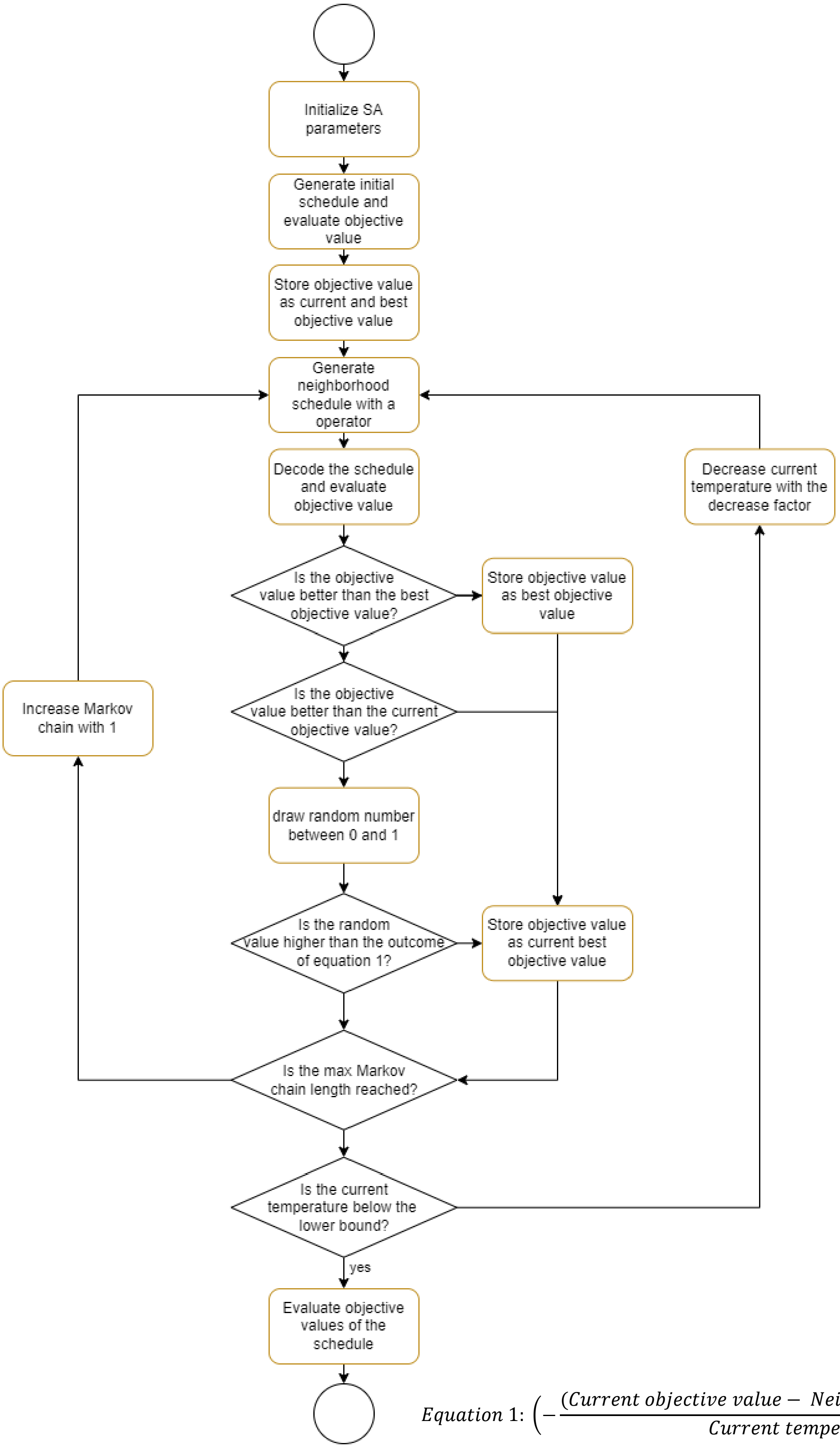
Adjusted NEH with Position Exclusion


List the jobs based on the NEH priority rule
Select first job of the list
Select the first stage from stages to schedule list
Select the first workstation in the stage
Determine the position of the job in the job priority list and on the sequence
List job to schedule and scheduled jobs based on the job priority rule
Create al list with scheduling positions based on the Evaluation Radius
Select the first position in the list
Schedule the operation in the position
Evaluate objective value of the schedule
Best objective value?
yes
Store objective value and position
no
Are all potential positions tried?
yes
Schedule the operation in the best position
Are all stages of the job scheduled?
no
yes
Are all jobs scheduled?
yes
Evaluate objective values of the schedule
remove job from list
remove position from list
remove stage from list

Simple Improvement Heuristic


Generate initial schedule and evaluate objective value
Store objective value as best objective value
Generate neighborhood schedule with a operator
Decode the schedule and evaluate objective value
Is the objective value better than the best objective value?
yes
Store objective value as best objective value
no
Is the maximum running time reached?
no
yes
Evaluate objective values of the schedule

Simulated Annealing


Initialize SA parameters
Generate initial schedule and evaluate objective value
Store objective value as current and best objective value
Generate neighborhood schedule with a operator
Decode the schedule and evaluate objective value
Decrease current temperature with the decrease factor
Is the objective value better than the best objective value?
Store objective value as best objective value
Is the objective value better than the current objective value?
Increase Markov chain with 1
draw random number between 0 and 1
Is the random value higher than the outcome of equation 1?
Store objective value as current best objective value
Is the max Markov chain length reached?
Is the current temperature below the lower bound?
yes
Evaluate objective values of the schedule

$$Equation\ 1{:}\ \left(-\frac{(Current\ objective\ value - \ Neighbor\ opbjective\ value)}{Current\ temperature}\right)$$