8/31/2020

Bachelor thesis

Dynamic scheduling using dispatching rules in a hybrid flow shop with stochastic processing times

Bastiaan Kauffeld

Industrial Engineering and Management
University of Twente

This report in intended for my supervisors at both the company for which this research is conducted, and the University of Twente.

University of Twente,
Industrial Engineering and Management
Postbus 217
7500 AE Enschede
Tel. (053)4 89 91, 11

Dynamic scheduling using dispatching rules in a hybrid flow shop with stochastic processing times

Author:

B.G. Kauffeld

Supervisors University of Twente:

First supervisor : E. Topan
Second supervisor : A.G. Leeftink

Date of publication: 31 August 2020
Number of pages excluding appendices: 45
Number of pages including appendices: 70
Number of appendices: 7

This report was written as part the bachelors thesis of the program Industrial Engineering and Management at the University of Twente.

Preface

Dear reader,

This research is part of finishing my bachelors program, Industrial Engineering and Management, at the University of Twente. I conducted this research in cooperation with a company, that has chosen to remain anonymous.

Before you start reading, there are a couple of people that I would like to thank. First of all, I would like to thank the company that I worked with, specifically my two supervisors at the company, whose names remain anonymous. Thank you for all your time, feedback and enthusiasm towards my thesis.

Secondly, I would like to thank my supervisors from the university, Engin Topan and Gréanne Leeftink. Your guidance and feedback has helped me to improve my research, as well as improve myself as a researcher.

Lastly, I would like to thank my family and friends, for showing their interest and supporting me where possible.

Bastiaan Kauffeld, August 2020

Management summary

We perform this research as part of a bachelor thesis for the study program Industrial Engineering and Management, at the University of Twente. The research is conducted at an external company, which chose to remain anonymous and will be referred to as the manufacturer.

Problem definition

In our research the focus is on an assembly process at the manufacturer. The assembly process consists out of two stages, with at each stage a number of identical and parallel workstations. Each workstation has its own inventory spot, where there is space for one job. There are two buffers, one at the front of the process and one between stage 1 and stage 2. This type of process is known as a hybrid flow shop (or flexible flow shop).

The problem that the manufacturer is facing, is having to flexibly switch the number of active workstations at each stage, as a result of starved or blocked workstations. This switching behavior leads to inefficiencies, resulting in the following choice of action problem:

"There should be a decrease of 99% in unplanned switching of workers in the assembly line at the manufacturer."

To contribute to solving this action problem, we identify the following core problem:

"Scheduling jobs to workstations takes place based on intuition, while we should make a decision based on available information."

Due to the stochastic processing times and release dates we choose to approach the problem with dynamic scheduling. This leads to the following research objective:

"Propose a dynamic scheduling strategy for the assignment of jobs to workstations in the assembly line at the manufacturer."

Research method

In a literature review we identify possible strategies for scheduling jobs to workstations at the assembly line. We make a selection of the most relevant strategies and use discrete event simulation to compare them. We use a total of 8 KPIs to compare the strategies. The two most important KPIs are the maximum tardiness and the makespan. The second most important two KPIs are the average tardiness and the percentage of late jobs. The least important KPIs are the average throughput time, average flow time, the number of blocks, and the number of starvations.

We use a total of 4 simulation experiments. In experiment 1 and 2 we find out the scores on KPIs of the dispatching rules. In experiment 3 we investigate the effects of decreasing the variability in deviations from estimated and actual completion times. In the last experiment, we investigate the effects of using different dispatching rules at stage 1 and stage 2.

Results

Based on literature, we identify that an event driven policy, using complete rescheduling, is the most suitable strategy. This means every time there is a new job arrival at one of the buffers or a workstation becomes empty, a new schedule is constructed. The most suitable method to construct a schedule, for the assembly line at the manufacturer, is using dispatching rules. The main results of the comparison of dispatching rules are shown in Table 1.

Table 1: Averages on KPIs from experiment 1, with colors indicating the relative performance.

KPI	EDD	MS	FCFS	ATC	PT+SL	SPT	Random
Makespan	703,98	667,00	716,81	739,61	764,88	796,88	718,02
Max tardiness	274,86	239,18	300,78	317,77	353,11	590,66	439,36
Avg tardiness	39,60	83,76	41,26	50,43	42,66	41,21	61,56
Percentage late	49,78%	86,12%	40,73%	75,22%	29,91%	26,72%	42,07%
Avg throughput time	105,88	119,78	105,98	119,91	114,32	114,41	107,66
Avg flowtime	244,85	314,75	227,35	279,13	191,20	178,82	231,49
Nr of blocks	0,05	8,35	0,35	1,95	6,65	7	0,35
Nr of starvations	62,00	44,50	64,35	32,00	61,85	48	58,95

The comparison of dispatching rules shows that EDD and FCFS outperform the other dispatching rules in most KPIs. MS performs best on the most important KPIs, the makespan and maximum tardiness. While the random rule scores roughly average on most KPIs, the variance of the KPIs are high, making the random rule less robust.

The result of experiment 3, where we investigated the effects of variability in completion times, shows that are no significant changes in relative performance, when comparing to the results of experiment 1. Experiment 4 shows that using different dispatching rules at stage 1 and stage 2 does not have a significant effect on the performance on KPIs.

Conclusion and recommendations

From the experiments we can conclude that the EDD rule and the FCFS rule has the best overall performance. However, the MS rule scores the best in the two most important KPIs, the maximum tardiness and the makespan. The SPT is only a good option for achieving low throughput times, flow times and percentage of late jobs. The PT+SL shows a good score in percentage of late jobs and a decent overall score, but is generally outperformed by the EDD and FCFS when it comes to overall scores. The ATC is by far the worst rule.

We recommend to use an event driven policy, with complete rescheduling. For the method of constructing a schedule we recommend the MS rule. While the MS rule scores relatively poor in the average tardiness, the percentage of late jobs, the average throughput time and the average flowtime, we still recommend this option due to its superior performance in the most important KPIs. The expected improvement of makespan is 5,3% in comparison to EDD, 7,0% in comparison to FCFS, and 7,1% in comparison to the random rule. The expected improvement of the maximum tardiness is 13,0% in comparison to EDD, 20,5% in comparison to FCFS, and 45,6% in comparison with the random rule.

Because of the timeframe of this research, we recommend the manufacturer to perform more tests on suitable methods for constructing a schedule. Specifically, we recommend to collect more input data for the simulation model, and testing more dispatching rules.

Contents

List of figures

List of tables

List of abbreviations

RQ	Research Question
MPSM	Managerial Problem Solving Method
HFS	Hybrid flow shop
HFSP	Hybrid Flow shop Scheduling Problem
MIP	Mixed Integer Programming
ATC	Apparent Tardiness Cost
EDD	Earliest Due Date
FCFS	First Come First Served
MS	Minimum Slack
PT+SL	Processing Time + Slack
SPT	Shortest Processing Time

1. Introduction

In the first chapter, we give an introduction to our research. Starting with an introduction to the problem owner, in Section 1.1. Then in Section 1.2 we give a brief context description. Lastly, in Section 1.3 we explain the research objective, research questions, and the research approach.

1.1 Introduction to the manufacturer

The problem owner of the problem that is presented in this research is an external company, at which we conducted our research. The company chose to remain anonymous, for that reason we refer to this company as the manufacturer.

1.2 Context description

The assignment proposed by the manufacturer for this bachelor thesis was to develop a strategy for assigning products to workstations, also known as scheduling, in one of their assembly lines. This assembly line can be classified as a hybrid flow shop (HFS), with two consecutive stages. In this process, products are transported from a buffer to the first stage, where they are processed, and afterwards transported to the second stage. After stage 2 is finished, the products are transported outside of the process. The assembly line is discussed in detail in Section 2.1.

The market the manufacturer operates in can be classified as "engineering to order", meaning the customer specifies the details of the products before they are made. As a result, almost every product that is processed is different and usually workers have not worked with the specific product before. The variety of different products makes it challenging to estimate the completion times, as well as cause a lot of variability in completion times. Apart from variety of products, more factors that cause poor estimation and variability in completion times exist. An obvious factor is, in contrary with machines, the work rate of workers performing manual labor varies. Other causes are incorrect supply of material to workers and errors caused by workers. As a result, planning and scheduling are a challenging for the company. The choice of which product to process next influences the performance of the assembly line, resulting in a loss of efficiency and tardiness of jobs. For example, if several products with long processing times are assigned to workstations in stage 1 at the same time, then this can result in idle time at stage 2. This means the workers in stage 2 become idle, because the flow of products has stopped.

The loss of efficiency and tardiness of jobs, is where the need of scheduling (sometimes referred to as sequencing) comes in. Scheduling is defined in the literature as the allocation of resources to jobs, on a certain time horizon, while optimizing one or more objectives (Pinedo M. , 1995). Scheduling activities typically result in a production schedule, containing information on when to process which product on which machine. An example of a schedule is given below, in Figure 1.

Figure 1: Example of a schedule, where 8 jobs are scheduled on 3 machines.

However, classical scheduling methods cannot be applied directly in this case, because classical scheduling theory focusses on deterministic cases, relying on the assumption that process times and release dates etcetera are known beforehand. In this case we have stochastic process times and stochastic release dates, making a predetermined schedule unsuitable. Therefore, we have a need for dynamic scheduling, which is an area within scheduling theory that focusses on scheduling in stochastic environments.

Right now, the company has no clear strategy when it comes to scheduling, meaning there has been little consideration on what an appropriate production sequence can be. Therefore, the manufacturer wants to know the possibilities of scheduling and what a suitable scheduling strategy would be for them.

1.3 Research approach

In this section, we explain the approach we used during this research by stating the research objective and research questions. An overview of the approach we used is given below, in Figure 2

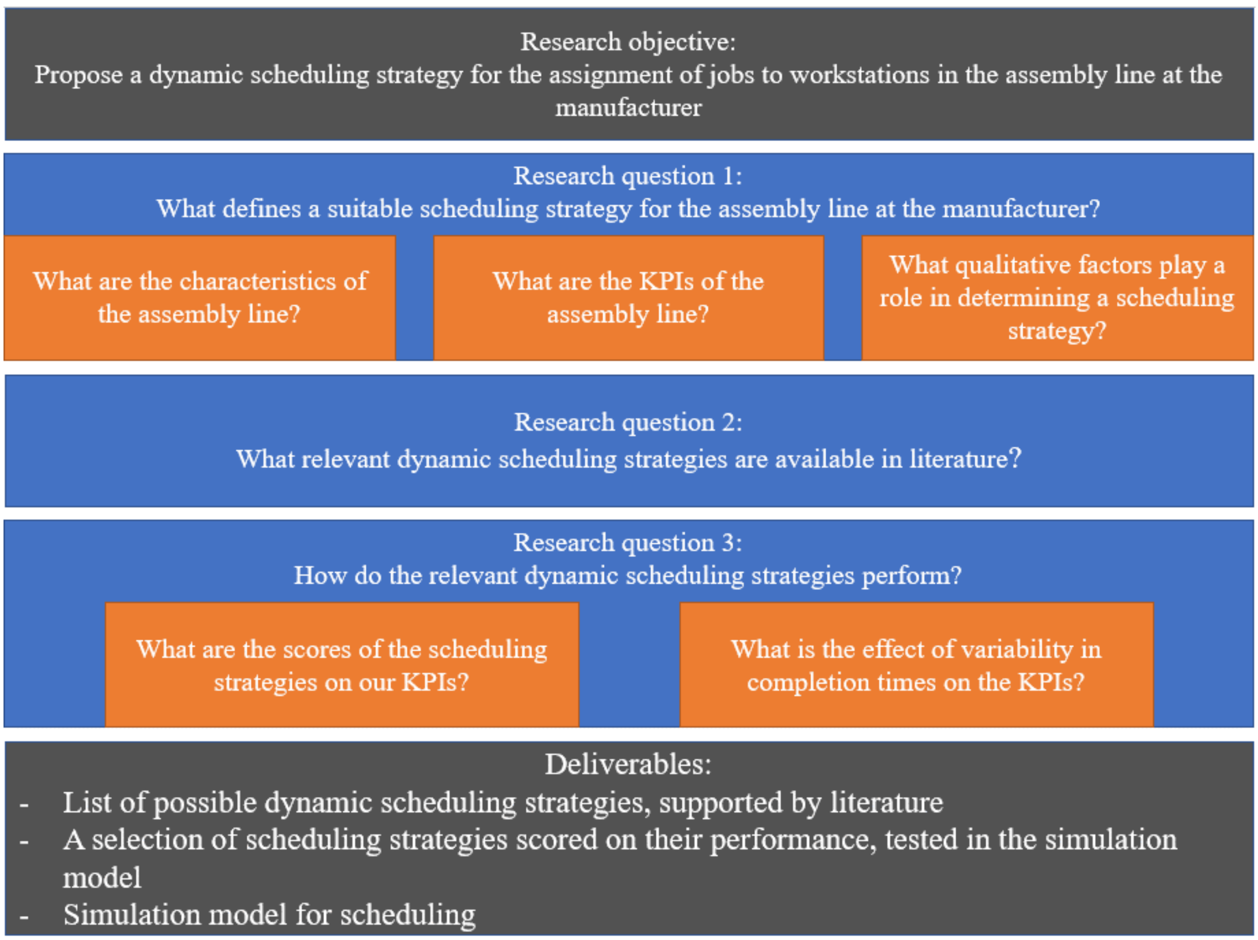

Figure 2: Overview of our research approach, with the objective, research questions and deliverables.

1.3.1 Research objective

The main research objective is as follows:

"Propose a dynamic scheduling strategy for the assignment of jobs to workstations in the assembly line at the manufacturer."

To achieve this goal, we have the following deliverables:

1) A list of possible dynamic scheduling strategies, supported by existing theory.
2) Simulation model for scheduling
3) A selection of scheduling strategies scored on their performance, tested using a simulation model.

1.3.2 Research questions

To reach the research objective we divided our research into research questions. We use this section to explain each research question (RQ).

1) What defines a suitable scheduling strategy for the assembly line at the manufacturer?

The first RQ serves to define what a scheduling should like for the assembly line. We want answers to this question, because the wide availability of methods for approaching scheduling problems forces us to narrow down our search. To answer this question, we have divided RQ 1 into three sub questions. We answered these questions by using observations and interviews with stakeholders at the manufacturer.

1a) What are the characteristics of the assembly line?

Before making any decisions on scheduling strategies, we want to know the details of the assembly line. We answer this question in Section 2.1.

1b) What are the KPIs of the assembly line?

Before we can say anything about the suitability of a scheduling strategy, we need indicators that measure the performance of scheduling strategies. Having clear objectives helps us in narrowing down our search for potential scheduling strategies, as well as compare them on performance. We discuss the KPIs in Section 2.4.1.

1c) What qualitative factors play a role in determining a scheduling strategy?

Apart from quantified and easily measurable KPIs, qualitative factors also play a role in choosing a suitable scheduling strategy. We discuss them in Section 2.4.2

2) What relevant dynamic scheduling strategies are available in literature?

Using the answers from our first research question we perform literature research to give an overview of available methods for scheduling in a dynamic environment. From these available methods we make a selection of strategies that are most relevant to our needs.

3) How do the relevant scheduling strategies perform?

After establishing a list of scheduling strategies that are worth investigating further, we use a simulation model of the assembly line to compare the different strategies.

3a) What are the scores of the scheduling strategies on the KPIs?

By answering this question we can make a distinction between the performance of different scheduling strategies, which helps us in determining what strategy is most suitable. The results are shown in Chapter 5.

3b) What is the effect of variability in completion times on the KPIs?

In addition to simulating the effects of different scheduling strategies on the performance of the assembly lines, we also want to gain insights on the impact of variability in completion times on the performance. For example, we want to know how more or less deviations from estimated completion times and actual completion times affect the scores of different scheduling strategies. We include this RQ, because the manufacturer aims to improve their predictions on completion times. Therefore, investigating these effects is relevant.

1.3.3 Methodology

In this research we follow the guidelines given in the Managerial Problem Solving Method (MPSM), a methodological approach often used in our field of study at the University of Twente.

The MPSM consists of the following phases (Heerkens & Van Winden, 2017):

1. Defining the problem
2. Formulating the approach
3. Analyzing the problem
4. Formulating (alternative) solutions
5. Choosing a solution
6. Implementing the solution
7. Evaluating the solution.

Given the timeframe and scope of our research, we do not consider phase 6 and 7 in our research. All other phases are visited during our research. In Section 2.2 and 2.3 we execute the definition of the problem. We then formulate the approach, which can be found in Section 1.3. Afterwards, the problem is analyzed throughout Chapter 2. Then we perform a literature review to formulate alternative solutions, in Chapter 3. Lastly, in Chapter 4 we explain the setup of our research experiments, and the results are given in Chapter 5, which form the basis for choosing a solution.

2 Context analysis

In the second chapter, we analyze the context further. We provide a process description in Section 2.1. Then in Section 2.2, we provide an overview of the related problems and summarize them in a problem cluster. In Section 2.3 we state the problem in terms of a core problem and an action problem. In Section 2.4 we define the relevant KPIs for the assembly line. In Section 2.5 we explain the research scope. We end the chapter by providing a conclusion, in Section 2.6.

2.1 Process description

In this section, we give an overview of the assembly line and elaborate on the different aspects of the assembly line.

The assembly line consists of two stages, which we refer to as stage 1 and stage 2, with multiple parallel and identical workstations at each stage. Each workstation has its own inventory, where there is room for one job. An overview of this process is given in Figure 3. In this process products are picked from a buffer, which we refer to as the front buffer, then transported to workstations at stage 1. When stage 1 is finished, jobs are brought to the next workstation where processing at stage 2 takes places. Both assembling operations are manually performed by workers. Between stage 1 and stage 2 there is a small opportunity for holding inventory. Note that there are two points at which we make a scheduling decisions. The first scheduling decision is at choosing a job at the front buffer, and the second is choosing a job at the intermediate buffer.

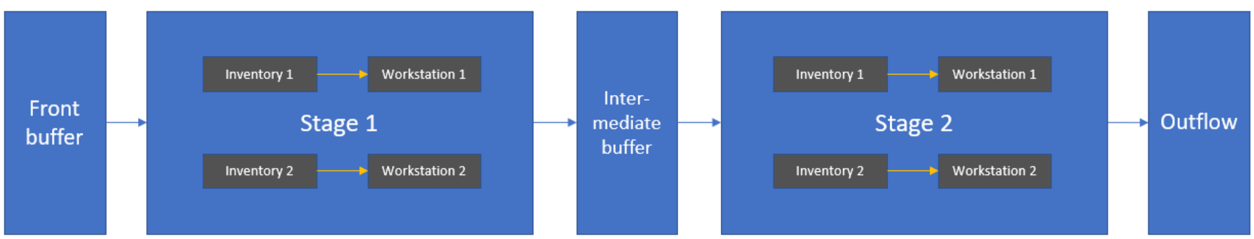

Figure 3: Overview of the assembly line

Processing times

The processing times of the jobs are stochastic, meaning they are not known beforehand. Estimates on completion times are made based on properties of the jobs, such as size, weight, complexity etcetera. In addition, the jobs have different processing times at each stage. Normally, the processing times at stage 1 are shorter than the processing times at stage 2. For this reason, the number of parallel workstations at stage 2 is higher than the number of parallel workstation at stage 1. Also, the processing times can vary a lot between jobs, from a couple of minutes to a couple of hours. Generally speaking, jobs that have a high processing time at stage 1, also have a high processing time at stage 2. More information on estimated and actual processing times is given in Appendix D: Input data, where we give an overview of the input data we used for our simulation model.

Release dates

The times at which jobs arrive at the front buffer, also known as release dates, are also stochastic. At the start of a working day a large portion of jobs are available for processing, which means they have a release date of 0. The rest of the jobs become gradually available over the day. Usually jobs are released in an interval of a few minutes.

Front buffer
Released jobs are stored at the front buffer. On average, about 100 jobs are available at the front buffer. The jobs that are available in the buffer form the set of available jobs to schedule to workstations at stage 1.

Intermediate buffer
Between stage 1 and stage 2 there is another opportunity to hold inventory and is used to maintain a high utilization of workers. When jobs are finished at stage 1, but the workstations at stage 2 are full, the jobs are first brought to this buffer. There is room for a maximum of 4 jobs at this buffer. The jobs that lie in the intermediate buffer form the set of available jobs to schedule to workstations at stage 2.

Due dates
The due dates of the jobs are based on planned departure times of trucks where finished jobs are stored. Therefore, most due dates of jobs come in groups, based on the truck they need to be in. For example, a set of jobs have the same due date. For each truckload there are always a few exceptions, some jobs have a due date slightly earlier than the rest of the truckload, for loading purposes.

2.2 Problem cluster

In this section, we give an overview of related problems that the manufacturer is facing at the assembly line. Afterwards we visualize the problems using a problem cluster and filter out potential core problems and highlight the action problem we are facing.

Blocking
One of the main problems the company faces are capacity issues, resulting in both overuse and underuse of the capacity. Overuse causes blocking in between stage 1 and stage 2. This occurs when the workers at stage 2 have too much work to handle, blocking the flow of jobs in the process. This leads to an overflow of the inventory between the stages. Blocking is costly, because the inventory it creates takes up valuable space, as well as manpower to handle the inventory. Also, severe cases of blocking lead to loss of production, since stage 1 workers cannot proceed working. Lastly, the accumulation of inventory that blocking causes, leads to higher throughput times of jobs. High throughput times lead to costs from a finance standpoint, because we already paid for the materials of the products, but we do not get our money before we deliver the products to the customer.

Starvation
The assembly process also deals with underuse of capacity, in terms of starvation for stage 2 workers. Starvation that workstations in stage 2 become idle, in case stage 1 workers do not finish their work in time, and there is no inventory left between the stages. Starvation is also costly, because of a loss of utilization, and consequently production. Between blocking and starving, the risk of starvation is higher, because the number of stage 2 working stations is generally higher than the number of stage 1 stations. As a result, a poor scheduling decision at stage 1 can result in starvation. For example, if we have 4 stage 1 workstations, and we schedule jobs with high processing times to each of them simultaneously, then stage 2 is very likely to suffer from starvation.

Switching workforce
The manufacturer solves the issues of blocking and starvation by switching workers reactively between different processes in the company, to maintain high utilization of workers. This switching behavior leads to inefficiencies, for starters the changeover times are high, also workers that are continuously switching make it harder to manage the processes effectively.

We should note that switching in itself can be useful, because it allows you to be flexible with your capacity. For example, if there are multiple jobs with high processing time that must go through the first stage, then temporarily increasing the capacity of stage 1 can be beneficial. However, in this case the switch is planned, in contrary to what happens in reality, where switching is used as a reaction to blocking or starving. Therefore, we introduce two kinds of switching, planned switching and unplanned switching, where only planned switching is wanted. At the moment switching takes place based on intuition.

In Figure 4 the problem cluster is given, which relates the problems that we identified in a causal chain.

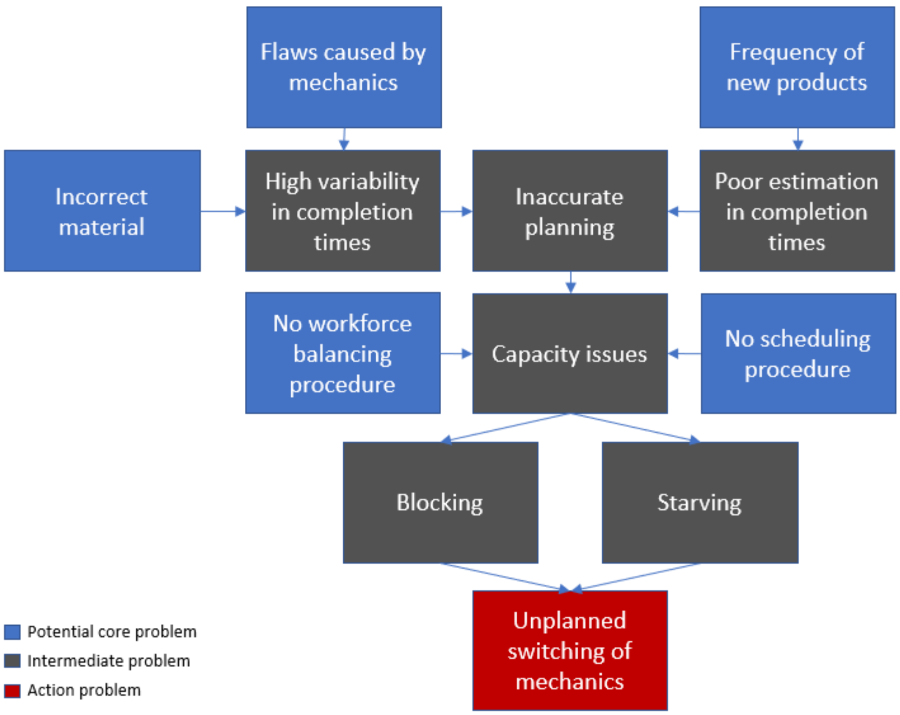

Figure 4: Problem cluster

2.3 Defining the problem

In the upcoming section, we start by discussing the action problem, afterwards we identify the core problem.

Action problem

To systematically solve our problem, we define an action problem and a core problem. As Heerkens & Van Winden (2017) explains, an action problem is a management problem, formulated as the discrepancy between norm and reality. The action problem that we identified for our case is formulated as follows:

"There should be a decrease of 99% in unplanned switching of workers in the assembly line at the manufacturer."

We realize this is an ambitious goal, and recognize that scheduling jobs to workstations alone will not solve this. However, the problem owner is striving to reach this goal. The scheduling strategy we provide should decrease the number of unplanned switches. In addition, the company is looking to implement more planned switches in the future, which can significantly decrease the number of planned switches as well.

The difference between a planned switch and an unplanned switch is whether you switch as a reaction, based on intuition, or switch in advance, based on the available information.

Core problem

The problem cluster in Figure 4 shows all problems that are connected to our action problem. The next step is to go back in the causal chain to find a core problem. We use the following rules of thumb in choosing a core problem (Heerkens & Van Winden, 2017).

1) Make sure that all problems in the cluster are related to the other problems. Also, we have to make sure the problem exists.
2) Potential core problems have no direct cause themselves.
3) If we cannot influence the problem, then it cannot become a core problem.
4) Choose the most important problem, one that has the greatest impact at the lowest cost.

The core problem we choose is:

"Scheduling jobs to workstations takes place based on intuition, while we should make a decision based on available information."

We explain the choice of our core problem in relation to the four conditions we mentioned. To ensure condition 1), we performed interviews and observation, and conclude that all problems exist and are related. Condition 2) is clearly fulfilled, because there is no direct cause to our core problem. Regarding condition 3), we are certain that we can influence our core problem, since we can provide a different scheduling strategy than the current one. One of the main reasons we choose scheduling is because it fulfills condition 4) quite well, namely the greatest impact at the lowest cost. We expect that providing a scheduling procedure based on scientific evidence is going to be an improvement on scheduling based on intuition.

2.4 Performance indicators

We use this section, to define the performance indicators of a scheduling strategy. Figure 5 contains an overview on the performance indicators we use. In Section 2.4.1 we start by defining quantitative KPIs and in Section 2.4.2 we define qualitative needs for a scheduling strategy.

KPIs	Qualitative factors
Makespan	Computation time
Maximum tardiness	Intuitiveness
Average tardiness	Applicability to other processes
Percentage of late jobs	
Average throughput time	
Average flowtime	
Number of blocks	
Number of starvations	

Figure 5: Overview of the KPIs and qualitative factors in determining a suitable scheduling strategy for the assembly process at the manufacturer.

2.4.1 Key performance indicators

The main idea behind introducing a scheduling strategy at the manufacturer is that a proper choice of job assignment to workstations decreases the likelihood of having to switch reactively to blocking and starving. However, scheduling has an influence on more factors than only blocking and starving. For example, we could schedule our jobs in such a way that blocking and starving rarely occurs, but at the cost of not meeting our due dates. Therefore, we do not measure the performance of a scheduling strategy by the number of switches, or number of blocking and starving events, but we use performance indicators that capture the most important factors such as due dates and the time at which all jobs are completed. Combining these objectives should decrease intermediate inventory, and prevent blocking and starving to occur, while also keeping due dates in mind (Weng, Wei, & Fujimura, 2012).

The first three KPIs capture the due dates aspect. The first KPI we introduce is the percentage of late jobs and is calculated as follows:

$$Percentage\ of\ late\ jobs = \frac{Number\ of\ due\ dates\ not\ met}{Total\ number\ of\ jobs\ processed}$$

Apart from wanting to know how many jobs are late, we want to know how much time they exceed their due dates. For example, if 20% of the jobs are late, but they are only late by 1 minute, then that would be better than 10% jobs that are late, but they are late by 1 hour. To capture this, we introduce the tardiness of a job, which is defined as $\max(0, Completion\ time - Due\ date)$, or in other words the number of time units the completion time of a job has exceeded its due date. We then calculate the second KPI in the following way:

$$Average\ tardiness = \frac{Sum\ of\ tardiness\ of\ all\ jobs}{Total\ number\ of\ jobs\ processed}$$

Apart from the average, we are also interested in the maximum tardiness, as this becomes an important KPI because we are dealing with truckloads. For example, if all jobs in one truckload are on time, except for one job which is late for 3 hours, then all jobs of that truckload are late for 3 hours, since the truck cannot leave without all its jobs. We calculate the third KPI in the following way:

$$Maximum\ tardiness = max\ (Tardiness\ job\ 1, Tardiness\ job\ 2, \dots\dots, Tardiness\ job\ n)$$

Apart from due date related objectives, we also have the objective of finishing all jobs as soon as possible. This objective is known as the makespan, and is defined as follows:

$$Makespan = \max(Completion\ time\ job\ 1, Completion\ time\ job\ 2, \dots\dots, Completion\ time\ job\ n)$$

Which comes down to the latest completion time of all jobs, or the time at which all jobs are completed.

The next two KPIs are of less importance than the first four, but instead give us more insights in what time jobs spend in the entire process. The next two KPIs are the average throughput time and the average flowtime.

We define the throughput time of a job as the time between the moment the job arrives at stage 1 and the moment the job leaves the process after being processed at stage 2. We calculate the average throughput time in the following way:

$$Average\ throughput\ time = \frac{Sum\ of\ throughput\ time\ of\ all\ jobs}{Total\ number\ of\ jobs\ processed}$$

We define the flowtime as the time between a job is released and the moment a job leaves the process. We calculate the average flowtime as follows:

$$Average\ flowtime = \frac{Sum\ of\ flowtime\ of\ all\ jobs}{Total\ number\ of\ jobs\ processed}$$

While we are still interested in this information, the actual outcome of these KPIs are less valuable in terms of performance, because a scheduling strategy with a high flowtime and throughput time can still perform well in terms of due dates and makespan, which are more important.

The last two KPIs we measure are the number of blocks at stage 1 and the number of starvations at stage 2. As mentioned before, blocking and starving are less important KPIs, since we do not care if there is blocking or starving, as long as this leads to good scores on more important KPIs. The reason we measure them is to gain more insights and because blocking and starving are related to our action problem of reducing the number of unplanned switches.

2.4.2 Qualitative needs

Apart from the KPIs we use, we also take into account some measurements for more qualitive performance of the different scheduling strategies. These measurements are no hard constraints, but instead they serve as guidelines to keep in the back of our minds when comparing different scheduling procedures. The aspects we discuss here mainly serve to improve the implementation of the scheduling strategy.

Computation time

An important aspect of a potential scheduling strategy is the computation time. As mentioned before, an optimal schedule would, in most cases, take too long to calculate, and therefore we use heuristics or simplifications to speed up the computation time. Also in our case at the manufacturer, computation time plays a big role, especially when we make use of dynamic scheduling, where new schedules are calculated on a regular basis. If, for example, we want to calculate a new schedule every 30 minutes, but calculating a new schedule already takes 40 minutes, then the schedule is of little use. Also, a schedule is based on the current state of the system. The larger the computation time, the more the current state of the assembly line has changed from the state at the moment we started computing. Within this trade-off between accuracy and computation time we are seeking to have an as accurate schedule as possible, within reasonable computation time.

Intuitiveness

Another factor influencing the rate of successful implementation is the intuitiveness of the scheduling strategy. Intuitiveness becomes a relevant subject, since there is a variety of complexity in existing scheduling methods. The higher the intuitiveness of the scheduling procedure, the higher the chances are of a successful implementation. The reason is that employees of different organizational layers, with different backgrounds, are going to be influenced by the scheduling procedure. Without trust in the scheduling strategy, employees might question the outcome, resulting in resistance from employees.

Applicability to other processes

The last factor that we include is applicability to different processes, which comes down to what extent the scheduling procedure (or parts of) can be used for different processes as well. For example, a scheduling strategy can be tailored to only one type of process, but a scheduling procedure can also be more general and applicable to multiple processes. Again, this is no hard constraint, but only serves as a guideline. If, for example, a scheduling procedure with low applicability to other processes performs exceptionally well, then we can still choose it over others.

2.5 Scope

Due to the limited time available for our research, we have to define our scope. The scope that we consider in our research is scheduling jobs to workstations in the assembly line at the manufacturer. We do not consider other processes at the manufacturer, or processes outside the manufacturer. Also, we limit our tests in the simulation model to a small set of scheduling strategies, because the time frame does not allow testing a wider range of strategies. In addition, many related topics such as workforce balancing or transportation utilization are relevant for the performance of the assembly line, but for simplicity reasons, we do not incorporate them into our research.

2.6 Conclusion

In this chapter, we answered RQ 1:*"What defines a suitable scheduling strategy for the assembly line at the manufacturer?* In Section 2.1 we started with answering the first sub research question: "*What are the characteristics of the assembly line?"* We answered this question by providing an overview of the assembly line and listed the characteristics afterwards.

In Section 2.2 we provided an overview of all related problems in a problem cluster. Based on the problem cluster, we determine an action problem and a core problem, in Section 2.3. We chose our action problem to be: *"There should be a decrease of 99% in unplanned switching of workers in the assembly line at the manufacturer."* Afterwards we chose a core problem: *"Scheduling jobs to workstations takes place based on intuition, while we should make a decision based on available information."*

We used Section 2.4 to answer the remaining two sub questions of RQ 1. In Section 2.4.1 we gave answer to RQ 1B: *"What are the KPIs of the assembly line?"* A total of 8 KPIs were formed, from which the maximum tardiness and the makespan are the most important. Less important are the average tardiness and the percentage of late jobs. The least important KPIs are the average throughput time and the average flow time, and the number of blocks and starvations. In Section 2.4.2 we answer RQ 1C: "*What qualitative factors play a role in determining a scheduling strategy?"* We recognized intuitiveness, computation time and applicability on different processes as the qualitative factors.

In Section 2.5 we set out the scope of our research.

3 Literature review

In Chapter 3, we answer RQ 2, in which we are looking for relevant scheduling strategies in the literature. We start in Section 3.1 by explaining the theoretical framework that is used during this research. In Section 3.2 we provide general approaches that the literature provides. In Section 3.3 we provide an overview of available methods for constructing schedules. We use the last section of this chapter, Section 3.4, to provide a conclusion on the literature review.

3.1 Theoretical framework

In the following section, we establish a theoretical framework in which we define the relevant concepts related to dynamic scheduling.

3.1.1 Hybrid flow shop scheduling problem

As mentioned before scheduling is known as assigning jobs to resources, in such a way that one or more objectives are minimized or maximized. The hybrid flow shop scheduling problem (HFSP) is defined in the following way (Fernandez-Viagas & Framinan, 2020):

We consider a set of jobs $N = \{1, \dots, n\}$ that are processed in a set of stages $S = \{1, \dots . s\}$. Each stage $i \in \{1, \dots s\}$ has m_i identical parallel machines. The processing time of each job $j \in \{1, \dots, n\}$ on a machine of stage $i \in \{1, \dots s\}$ is given by p_{ij}. In addition, all jobs follow the same sequence of stages, and machines can only handle one job at a time. Regarding set up times, we either neglect them, or incorporate them into the processing times. We denote the completion time of job $j \in \{1, \dots, n\}$ in stage $i \in \{1, \dots s\}$ by C_{ij}.

3.1.2 Conceptual model

As part of our theoretical framework we define concepts that are relevant to *dynamic scheduling*. An overview of the concepts is given in a conceptual model, shown in Figure 6.

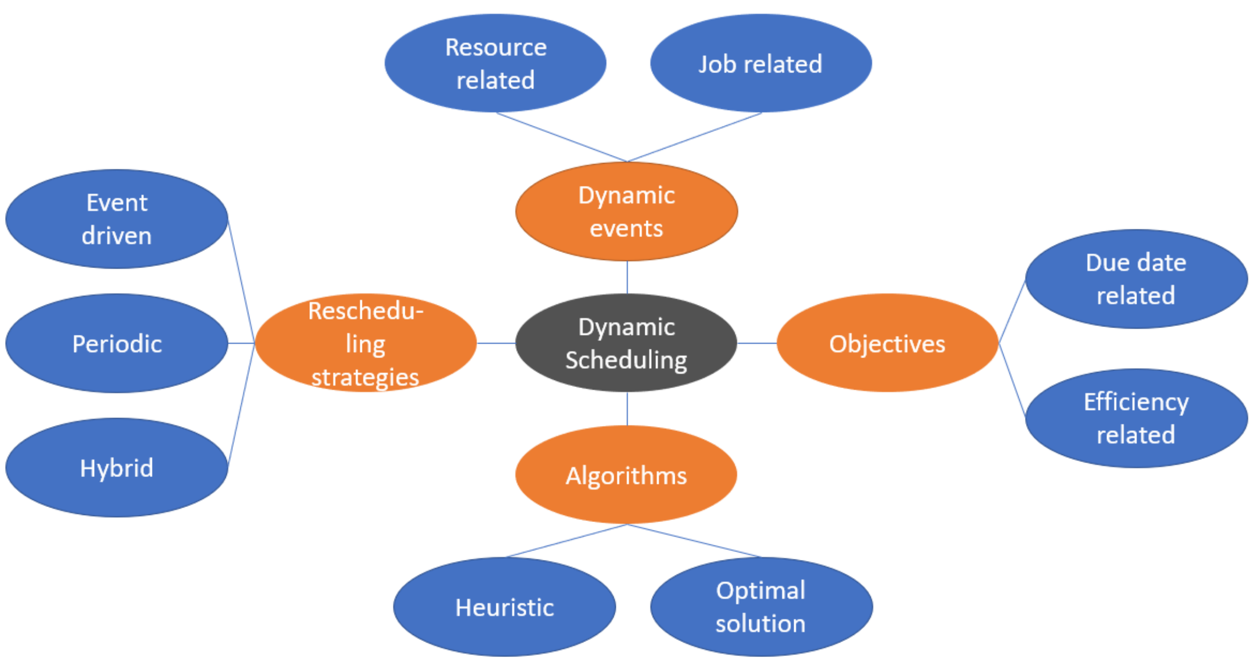

Figure 6: Conceptual model of dynamic scheduling.

Dynamic events
The main difference between dynamic scheduling and classical scheduling is the presence of *dynamic events*. The events that are related to dynamic scheduling can be classified into two categories: job-related and resource related (Chen, et al., 2019). Examples of job related events are changes in due dates, completion times, or emergency jobs. Resource related examples are breakdowns of machines or operator illness.

Rescheduling
To cope with the dynamic events, *rescheduling* is often used. The literature provides two main rescheduling methods, namely schedule repair and complete rescheduling (Ouelhadj & Petrovic, 2009).

Rescheduling strategy
The frequency at which rescheduling takes place is often referred to as the *rescheduling strategy*, in which three main strategies can be identified. The first strategy is periodic rescheduling, which divides the dynamic problem into sub problems that can be considered as regular scheduling problems (Chen, et al., 2019). The period determines the frequency of which a deterministic schedule is updated. Another strategy is event driven scheduling. An example is multi-agent based dynamic scheduling (Shi, Guo, & Song, 2019), this approach updates a schedule when disruption events occur. The last type of strategy is a hybrid between the first two strategies, where the schedule is either updated on a periodic basis, or if significant disruptions have taken place.

Scheduling algorithms
Solving the HFS dynamic scheduling problem is done by either an exact algorithm or a heuristic (Weng, Wei, & Fujimura, 2012). Exact solutions are proven to be strongly NP-hard, meaning computation time takes considerable time. Therefore, heuristics are often used in practice. Heuristics are efficient models that find near-optimal solutions within reasonable computation time.

Objectives
The goal of dynamic scheduling is to find the optimal schedule according to one or more *objectives*. Typical objectives that occur in a HFS are related to due dates, earliness and tardiness (Weng, Wei, & Fujimura, 2012). In addition, efficiency related are often included as an objective, like minimizing the makespan.

3.2 General approach to dynamic scheduling
In this section, we elaborate on the theory on dynamic scheduling in a HFS, based on literature. We start by mentioning a couple of general points regarding dynamic scheduling. Afterwards, we distinguish three general approaches to handling stochastic factors in scheduling. Next, we explain the difference between complete rescheduling and scheduling repair, which are the two main ways to handle the HFSP.

General points on dynamic scheduling
While elegant mathematics and algorithms exist to solve classical scheduling problems in simple machine environments, the complexity quickly rises when considering more complex machine environments, such as the HFS (Pinedo M. L., 2016). Solving the deterministic case of the HFSP is proven to be strongly NP-hard, making exact solutions not a viable option, as their computation time is very high. In the stochastic case this complexity only rises. The literature agrees that a gap exist between theory and practice. Pinedo (2016) argues that the gap exists, because the simplistic models used in theory are usually very different than the complex manufacturing systems in practice. He gives a variety of reasons, from which we list a few that are relevant when we apply theory to our case at the assembly line:

1) Theory often assumes that all jobs to be scheduled are known beforehand, while in reality jobs often become available continuously, subject to some degree of randomness.
2) A lot of theory does not emphasize on rescheduling, neglecting the fact that revision of a schedule might be beneficial, due to stochastic events.
3) Real manufacturing systems differ a lot from theoretical models, specifically real manufacturing systems have more restrictions than literature takes into account.
4) Most theory does not take into account that machine availability changes over time.
5) Stochastic models that are used in literature have specific assumptions. For example, some models assume that processing times have an exponential distribution. However, in reality processing time usually do not follow exponential distributions.
6) Processing time of jobs of the same type have the possibility of changing, due to learning curves, while the theoretical models often assume the processing times to be fixed.
7) In practice often seemingly cruel approaches are used for solving scheduling problems. Since the more randomness the process contains, the less sophisticated methods are applicable.

On the bright side, Pinedo (2016) states that time spend on theory was not wasted, because a lot of the methods that try to cope with dynamic environments are based on the theory that has been developed. In addition, he mentions that successfully applying scheduling theory to practice often requires a mix of the theoretical models and methods.

One of the gaps of existing research on dynamic scheduling is that it tends to focus on only one dynamic event (Peng, et al., 2019). For example, a research often takes into account only machine breakdowns, neglecting the fact that more dynamic events occur. Furthermore, a lot of research tends to use probability theory and fuzzy logic, but therefore lacks to cope with reacting to real-time events (Qin, Zhang, & Song, 2018). In addition, they mention that the research that does try to cope with reacting to real time events tends to focus on one dynamic event, mostly random job arrivals, rush jobs or machine breakdowns. This leaves uncertain processing times in dynamic scheduling still a comparatively limited subject.

Approaches to solving dynamic scheduling problems

The dynamic events that cause the need for revision of a schedule are dealt with by three major rescheduling strategies (Chen, et al., 2019), which are event driven, periodic and hybrid strategies. In most methods the dynamic scheduling problem is divided into static subproblems, as they are known in classical scheduling theory. In event driven strategies (sometimes referred to as online scheduling), every time an event occurs (e.g., a workstation completed a job), rescheduling takes places. This strategy is likely to give the best performance, however computation time can be a problem, because events can occur frequently. The second strategy, the periodic strategy, makes a new schedule every fixed period of time. The new schedule is executed until the end of the period, at which a new schedule is made. The last strategy is called the hybrid strategy, this strategy is a mix between periodic and event driven scheduling. An example of a hybrid strategy is that a new schedule is determined every fixed period, or earlier if major events causes the state of the system to change significantly.

With one of the challenges in applying scheduling theory in practice being that every process is completely different, it is not clear which of the three methods is the best for the assembly line at the manufacturer. One could argue that event driven scheduling might be necessary due to the high presence of stochasticity, but an immediate counter argument would be the threat of high computation time. However, the relatively high processing times of the jobs mean that the time between events is relatively high as well. This makes event driven scheduling an interesting option to investigate further. Also, the hybrid option is interesting, since a revision of the schedule might only lead to significant improvements if unforeseen events have taken place. The periodic approach is the simplest and would perform the best in terms of computation time and is easier to implement, however the threat exists that performance goes down in case of unforeseen events between periods of rescheduling. While exploring other options can be interesting for further research, we choose to only include event driven rescheduling in the simulation study. The main reason is the high presence of stochastic factors, as well as the fast computation time of the chosen scheduling algorithms, make the event driven strategy the most promising.

Rescheduling strategies

Once you have decided on how often you want to revise your schedule, a rescheduling strategy has to be chosen. The literature provides two main strategies, namely schedule repair and complete rescheduling (Ouelhadj & Petrovic, 2009). The schedule repair strategy applies local adjustments to the current schedule, based on the new information about the state of the system. These minor adjustments have two benefits, namely saving computation time and causing minimal disruption to the entire system. Disruption to the system can in some cases be relevant, if for example, a schedule for the entire day is used for different planning activities as well. Significant deviations from the initial schedule can result on negative implications on other planning activities. However, schedule repair does come at the cost of performance.

A common strategy that uses schedule repair is robust scheduling. In a paper that focusses on unknown release dates of jobs a robust scheduling strategy, using a hybrid approach, is proposed (Jianyu, et al., 2020). This paper predetermines a forecast of the job releases for a longer period of time, and makes a schedule based on the forecast. As the time elapses, they update the schedule if the actual job releases cause large deviations between the forecasted release dates and the actual release dates. When updating, minimizing the changes from the original proposed schedule is a part of the objective function.

The other strategy Ouelhadj & Petrovic (2009) mentions is complete rescheduling, which calculates an entire new schedule from scratch. While this can take significantly more time to compute and disrupts the entire system, the results should be better than using schedule repair.

In our case, schedule repair is less interesting, for a couple of reasons. First of all, we do not consider disruptions of an initial schedule to be of great importance. Secondly, with the high presence of stochasticity, an initial schedule is most likely to be inaccurate after processing times turn out to be different than expected. While schedule repair is interesting in terms of saving computation time, we think that the loss in performance makes schedule repair less interesting. This leaves us with the option of complete rescheduling, which is in addition to the mentioned downsides of schedule repair, also more applicable to the assembly line at the manufacturer, because complete rescheduling handles the changes in division of workforce more easily.

3.3 Overview of solving methods

In this section, we provide an overview of available scheduling algorithms for solving the HFSP, that can be applied within one or more of the approaches. For each of the methods we shortly discuss some of the benefits and downsides in relation to our case at the manufacturer.

(Composite) dispatching rules

According to Ouelhadj & Petrovic (2009), dispatching rules have played a big role in dynamic contexts, and a lot of simple and complex dispatching rules have been proposed by the literature. A big advantage of dispatching rules is that they are easy and intuitive, therefore easy to implement. However, a main drawback is that they usually perform poorly in comparison with more advanced methods, due to the myopic nature of dispatching rules. In addition, Pinedo (2016) points out that composite rules only focus on one objective.

Since the machine environments that are considered in practice often require multiple objectives, composite dispatching rules were introduced. These composite dispatching rules are more complex than the regular dispatching rules, because they consider multiple objectives. Pinedo (2016) gives an example of a commonly used composite dispatching rule, the Apparent Tardiness Cost (ATC). The ATC combines the Weighted Shortest Processing Rule with the MS rule

Due to their intuitiveness, fast computation time and easy implementation, we consider both dispatching rules and composite dispatching rules as a method worth investigating further. While the solution of the rules would not be near optimal, given the timeframe of this research and the needs at the manufacturer, dispatching rules are most promising. Therefore, we included several dispatching rules for comparison, in our simulation study. The choice of dispatching rules is given in Section 4.1.2.

Mixed integer programming
Constraint based programming techniques like mixed integer programming (MIP) are another way of solving the HFSP. While a lot of research on dynamic scheduling formulate their specific scheduling problem as a MIP model, they acknowledge that solving the model is often not possible within reasonable time. However, Pinedo (2016) does provide some methods, such as the branch and bound method, that can calculate suboptimal solutions to these MIP models.

Due to their complexity, high computation time and because each problem requires very specific formulations, we do not include MIP models into our simulation model.

Stochastic models
Another way of solving the HSP in dynamic environments is by using stochastic models. Pinedo (2016) explains that various methods exist that can be used to model the randomness that occurs in dynamic environments. These models use probability theory to predict certain outcomes like job completion times, based on their underlying distribution. An example is applying models that are used in queuing theory to study the behavior of a production environment. A big downside, as mentioned in Section 3.2, is that these models rely heavily on the underlying assumptions of statistical distributions. For example, queuing theory uses the assumption that processing times follow an exponential distribution, but we know that is almost never the case.

Since the performance of the stochastic models relies heavily on the assumptions on processing times, we conclude that they are not suited for the assembly at the assembly line. The main reason is that we know that the estimates we have on the processing times of jobs are often inaccurate, and are therefore not a good basis to form our decisions on.

Metaheuristics
A popular approach to solving production scheduling problems are meta-heuristics, which are higher level heuristics that guide local search algorithms away from local optima (Ouelhadj & Petrovic, 2009). Local search algorithms take a schedule and try to improve it via so called neighborhood operations. An example of a neighborhood operation is swapping two adjacent jobs in the schedule. The main drawback of local search algorithms is that they converge to what is called a local optima, from which no neighborhood operation can be applied anymore that improve the schedule. These local optima can be far away from

the global optimum, and therefore local search usually does not perform well enough. Metaheuristics look like local search algorithms, but try to escape from local optima by sometimes accepting bad solutions in an intelligent way. There is a wide majority of metaheuristics available, but the most mentioned algorithms in literature are simulated annealing, ant colony optimization and genetic algorithms.

The good solutions, wide applicability make metaheuristics an interesting method for solving our case. However, the main disadvantages are the required computation time and the complexity of the algorithms. Therefore, we do not include them in the simulation study.

3.4 Conclusion

We started this chapter in Section 3.1 by providing a theoretical framework, which we depicted in a conceptual model in Figure 6.

We used Section 3.2 and Section 3.3 to answer our second research question: "*What relevant dynamic scheduling strategies are available in literature?"* We broke a scheduling strategy down into a rescheduling strategy, a rescheduling frequency, and a scheduling algorithm. In Section 3.2 we provided a number of options, from which we chose complete rescheduling as our rescheduling strategy, and an event drive policy as our rescheduling frequency. We chose these, because they are expected to handle the stochasticity of the processing times and release dates in the best way.

In Section 3.3 we discussed various algorithms used for constructing a schedule. We chose dispatching rules to be most relevant to the assembly process, because of its simplicity and fast computation times. We did not choose one of the more advanced algorithms for constructing a schedule, because they rely more heavily on assumptions, like deterministic processing times. In addition, the complexity of the more advanced algorithms are a lot harder to implement, making them less suitable for the manufacturer.

4 Solution design

In this chapter, we explain how we apply scheduling in the assembly process and how we are going to compare different strategies using a simulation model. In Section 4.1, we explain the strategy that we to use in the simulation model and give an overview of the different scheduling algorithms that we test in the simulation model. We use Section 4.2 to elaborate on the simulation model that is used. In Section 4.3, we give a conclusion of the chapter.

4.1 Scheduling strategy

4.1.1 General strategy

The literature review has led us to use the following main strategy. We make choices on our rescheduling frequency, rescheduling method and ~~on~~ what scheduling algorithm to use for determining a schedule. First, we focus on the first two, the frequency and method of rescheduling. After discussing these two, we discuss how we take into account the inventory at the workstations into the scheduling strategy.

Event driven policy

The rescheduling frequency that we use is an event driven policy, meaning that certain events in the assembly line trigger the construction of a new schedule. The first event we choose are job arrivals, either at the front buffer, or at the intermediate buffer. The second event is after a workstation, or when an individual inventory of a workstation becomes empty. For example, after a job is finished on a workstation, the workstation is emptied.

The event driven policy, with the events that we choose, means that we are rescheduling frequently. We make this choice, rather than rescheduling periodically, for two reasons. First of all, the amount of stochasticity in the dynamic events make rescheduling periodically less interesting. For example, if we make a schedule now, then in 20 minutes the schedule is likely to be inaccurate due to new job releases and deviations in expected and actual completion times of jobs. The second reason is related to computation time, which is one of the main reasons to consider periodically, rather than on a very frequent basis. Since we only focus on dispatching rules, which are known for their fast computation times, we have the luxury to reschedule frequently.

Rescheduling method

When rescheduling, we have the choice to either make a new schedule from scratch, or to repair the last schedule based on the dynamic events that happened in the meantime. One of the main reasons for considering repairing schedules, is that it allows you to stick more closely to an initial schedule, which is often related to other planning and control activities within a process. Secondly, repairing a schedule saves significant computation time. In this study, we do not have the objective of sticking to an initial schedule and computation time is not a problem. Therefore, we focus only on complete rescheduling.

Leaving a workstation empty

An important aspect of having inventory spots at workstation is that we may not want to load these spots. Consider the following example; we load the inventory spot of workstation 1 with job x, while job y is in progress at workstation 1. We expected that job y is finished soon, but due to the stochasticity in the completion times, the job takes longer than expected. Meanwhile, workstation 2 becomes idle, because the job finished earlier than expected. Now we regret putting job x at the inventory spot of workstation 1, because we would rather have the job at workstation 2. Because of this example, choosing to wait with loading a workstation can be beneficial. However, the tradeoff is that there is a chance a job is finished early, in that case the worker becomes idle and has to wait until the workstation is filled. . In our simulation model we always fill the workstation, except for a special case. In this case the number of available jobs is small. If we make a schedule, but all the jobs are assigned to different workstations, because they all contain smaller jobs, then we leave the station empty. Whenever a new event occurs, as defined within our event driven policy, we make a new schedule and again assess if we want to fill the workstation or let it remain empty. In Appendix A: Flowcharts of events, we provide flowcharts that give an overview of the scheduling process.

4.1.2 Scheduling algorithms

In this section, we explain all the dispatching rules that we choose to incorporate into the simulation study for comparison. The selection we made is based on the popular dispatching rules in the literature.

Earliest due date (EDD)

This is, together with FCFS and SPT, one of the most classical and simple dispatching rules. The EDD rule orders all jobs on their due dates, so the job with the earliest due date is ranked first.

First come first served (FCFS)

The FCFS rule, which is in this case similar to the well-known First In First Out (FIFO) rule, orders all jobs according to their release date. This means the job with the earliest release date is ranked first.

Shortest processing time (SPT)

As its name suggests, the SPT ranks the jobs according to their processing time. The job with the shortest processing time is ranked first. The SPT rule is known to be good for the utilization of your process (Pinedo M. , 1995). However, because it does not take into account any due dates, the rule usually does not perform well towards due dates related objectives.

Minimum slack (MS)

The MS rule ranks the jobs according to their slack times. A slack of a job is defined by $\max(due\ date - remaining\ processing\ time - current\ time, 0)$. In other words, the remaining time until the job has to be started in order to reach its due date. With the MS rule, the jobs are ranked from least slack to most slack.

Processing time + slack time (PT+SL)

The next rule we use is a variation on the PT+WINQ+SL rule. The regular PT+WINQ+SL rule ranks beams by looking at their process time, slack time and takes into the work in the queue of the next operation.
In the variation we neglect the work in the queue of the next operation and pick the job with the minimum $processing\ time + slack\ time$. In contrary with the MS rule, here the we define the slack as $\min(due\ date - remaining\ processing\ time - current\ time, 0)$.

Apparent tardiness cost (ATC)

The ATC is a more complex dispatching rule, that ranks the jobs based on a combination of the SPT rule and the MS rule. The goal is to utilize the property of the SPT rule of high utilization, while keeping into account due dates.

Before giving the formula for the ATC rule, we explain the terminology used in this formula:

p_j : $Remaining\ processing\ time\ of\ job\ j$
$\bar{p}$: $Average\ processing\ time\ of\ remaining\ jobs$
d_j : $Due\ date\ of\ job\ j$
t : $Current\ time$
K : $Scaling\ parameter$

The formula for each ranking $I_j(t)$ of each job is as follows (Pinedo M. , 1995):

$$I_j(t) = \frac{1}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K\bar{p}}\right)$$

The larger K is, the more emphasis on the SPT rule. Then obviously, the smaller K is, the more emphasis on the MS rule. In case of the ATC rule, the job with the highest score is scheduled first.

With the ATC rule, jobs are scheduled one at a time, meaning every time a workstation becomes free again, a new ranking index is computed. This differs from the other rules, because the ranking of the jobs with the ATC changes over time, where for the other rules the ranking remains the same. For example, when we are assigning all of the jobs to a workstation, the ranking of the jobs at time t = 0 can be in a different order than when we are looking into the future, at for example time t = 10. This is caused by the exponential part of the formula, which does not behave linearly when t increases.

To determine the scaling parameter K in the formula, we performed a parametric analysis by running the simulation model for different values of K. This analysis is included in Appendix F: Parametric analysis ATC rule.

Random
The random rule randomly orders the available jobs. We use the random rule to compare the effects of using dispatching rules in comparison with using random scheduling. While in the current situation scheduling takes place based on intuition, the random procedure is likely to be the closest to the current situation.

4.2 Simulation model

In the following section, we elaborate on the simulation model we used for testing different solutions.

4.2.1 Discrete event simulation

The type of simulation used for our experiments is called discrete event simulation. This type of simulation is used to simulate processes by jumping from event to event skipping the time in between, rather than continuously simulating. An event triggers certain actions and often plans new events for the future. In our case we use events for arrival of jobs at buffers, completion of jobs, filling workstations, moving jobs, and events used for initialization.

The goal of our simulation study is to compare the different scheduling algorithms as proposed in Section 4.1.2. During the simulation run we collect statistics that we use to calculate KPIs. We use the KPIs that we mentioned in Section 2.4.1.

4.2.2 Conceptual model

A conceptual model is often used a bridge between the developer and the user or problem owner. In the conceptual model we explain what our simulation code should do. With our model we simulate the assembly process as depicted in Figure 3 from Section 2.1. However, the number of parallel workstations differs from what is in the figure.

Apart from a broad overview, we want to give more detail into the simulation from a conceptual perspective. Therefore, we used flowcharts to visualize the most important simulation events. To give an idea, we include one of the main events, namely when a stage 1 workstation is empty, shown in Figure 7. This event is triggered after a stage 1 workstation is emptied. We start by determining a new schedule, then we check if the workstation that was emptied in included in the schedule. If the workstation is not included, we change the status of the workstation to being starved. If the workstation is included then we update the workstation and remove the job we just scheduled from the front buffer. Lastly, we check whether the worker at the workstation is idle. If he is idle, we start the job and generate a completion event. If he is still working on another job, then we do nothing. We included the rest of the main events in Appendix B: Conceptual models for simulation model.

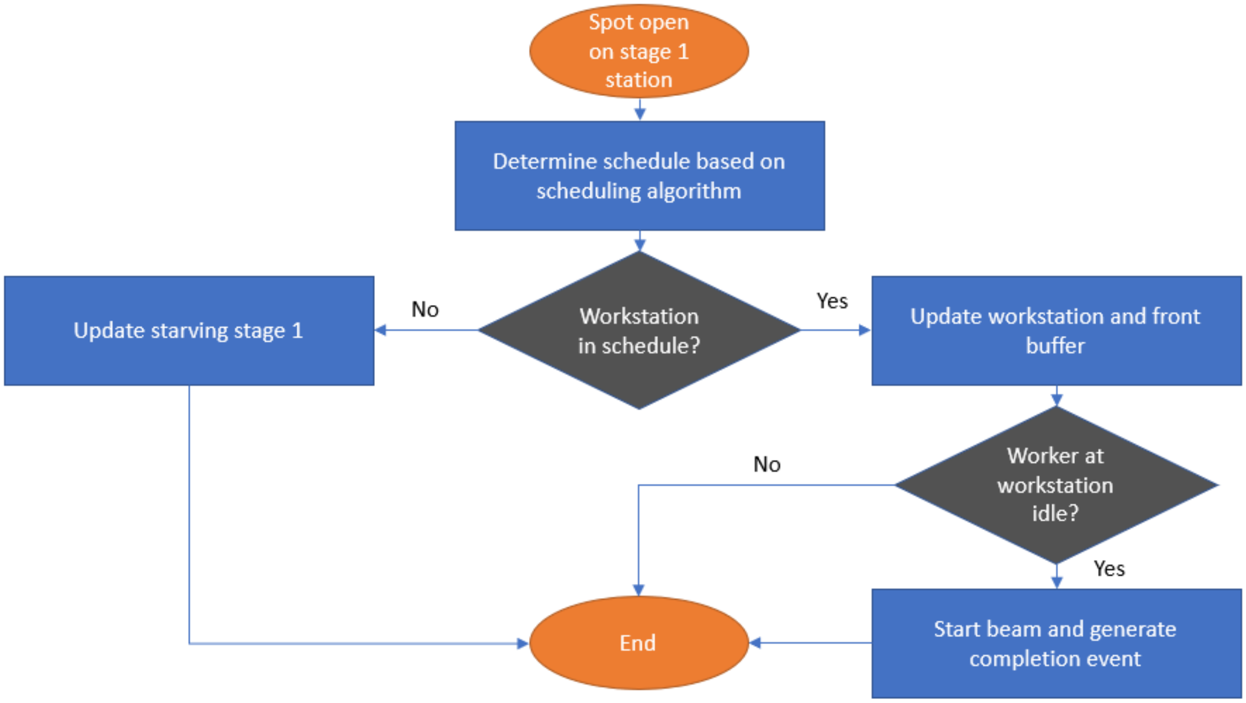

Figure 7: Flowchart of stage 1 workstation empty event.

Scheduling

The scheduling in the simulation model takes places according to the general strategy we discussed in Section 4.1.1. However, for practicability reasons we used a method that saves a lot of computation time. This method makes use of the fact that since we make a new schedule whenever we fill a workstation, we do need to know anything about jobs that are scheduled after we assign a job to the workstation we want to fill. To make use of this, we include an algorithm, shown in Appendix C: Algorithm used for scheduling in simulation model, that is triggered whenever we try to fill a certain workstation. First, we assign the jobs one by one to the workstation, until either we assign a job to the workstation we are trying to fill, or all available jobs are assigned. If the latter occurs, then we return that we do not want to schedule a job at this moment to the workstation we are trying to fill. However, there is one exception where we still want to fill a workstation, even if the schedule suggests that we shouldn't. This exception occurs when we have an empty stage 2 workstation and the schedule suggests that we should not fill it yet, but we are currently dealing with blocking in stage 1. In this case finished jobs from stage 1 cannot be moved towards the intermediate buffer, because the intermediate buffer is full. If this happens, we assign the job to the stage 2 station anyways, making room at the intermediate buffer for the blocked jobs from stage 1.

4.2.3 Input data

In order to simulate the assembly process we need input data, specifically on jobs that have to be processed, and the number of parallel workstations at each stage.

Jobs

Together with the manufacturer we established a set of jobs that provides an accurate representation of the assembly process. The jobs each have corresponding due dates, release dates, estimated completion times, and actual completion times. Because the amount of historical data on estimated and actual completion times is limited, we duplicated every recorded job once and shuffled them around a bit to remain close to the actual system. This dataset is included in Appendix D: Input data. This set of jobs contains 116 jobs and takes around 10 to 12 hours of total production time, depending on what scheduling algorithm is used. The release dates and the due dates of the jobs are very tight, as a result it is impossible to process all the jobs on time.

Additional input data

While the dataset described above best represents the actual system, the amount of data is limited. There is too much room for coincidence when it comes to scoring the different algorithms on performance, because the stochastic factors influence the performance of the scheduling algorithms. For example, for some of the jobs the actual completion times can differ by one hour from the estimate. If one of the algorithms scheduled one of these jobs at the very end, the makespan would be heavily affected. Therefore, we want to eliminate the luck factor as much as possible, making our result more viable. In order to do so, we perform multiple simulation replications using the same due dates, release dates and estimated completion times, but with different actual completion times, since this stochastic factor has the most influence on the performance. For each replications we determine the actual completion times by sampling a deviation percentage from the estimated completion times and add this to the estimate. We use a deviation percentage because a job with a higher completion time is more likely to have a higher absolute deviation, than a job with short completion time.

We determine the deviations by assuming a separate normal distribution for stage 1 and stage 2 and then take a random sample out of these distributions. The parameters, as well as upper and lower bounds we use for sampling from these distributions, are given in Table 2. We use bounds to prevent outliers on deviations. The establishment of the parameters for these normal distributions, as well as more information on the deviations, can be found in Appendix E: Choosing distributions and parameters for input data. We should note that we do not have convincing evidence that assuming a normal distribution is justified. But since our goal was to create deviations from the estimates, in such a way that we can simulate the effects deviations in completion times have on our chosen dispatching rules, we believe the assumption still provides us with what we need.

Table 2: Parameters and bounds for sampling deviation percentages for the completion times of stage 1 and stage 2 jobs.

Stage	Mean	Sigma	Lower bound	Upper bound
1	-8,64	53,85	-80,17	142,04
2	-1,19	48,72	-60,59	164,64

Number of parallel workstations

The number of parallel workstations at each stage is based on the workload of the jobs. For the simulation experiments we use 4 workstations at stage 1 and 13 workstations at stage 2.

4.2.4 Simulation runs

Number of replications

As we explained in the previous section, we use multiple replications of the simulation experiment. While making multiple replications increases the accuracy of our experiments to evaluate the different scheduling algorithms, it also increases computation time. Therefore, we perform 20 replications for each of the interventions in all 4 experiments.

Start and end of run

At the start of the simulation run, we start the way an actual working day in the assembly process starts. We begin by filling all stage 1 stations and let the workers start. The stage 2 workers start 1 hour later, because otherwise they would have to wait on jobs to finish in stage 1. The simulation run ends when all jobs have been processed. This means we do not use any warmup period.

4.2.5 Assumptions and simplifications

In order to simulate the assembly process, we have to make some assumptions and simplifications, because it is almost impossible to capture the real system in a simulation model. We make assumptions when we have certain gaps of knowledge that we have to fill. When the real system is too complex, we make simplifications, because it would be impossible to simulate the system otherwise. We make the following assumptions and simplifications:

List of assumptions:

- Deviation percentages from actual and estimated completion times are normally distributed.
- Whenever more workstations are blocked at stage one and a spot opens at the intermediate buffer, we choose to solve the blocked workstation that has been blocked the longest.

List of simplifications:

- All parallel workstations at a stage are identical, meaning they have the same work rate and can handle any kind of job.
- All transportation of jobs takes the same amount of time.
- The amount of parallel workstations at each stage remains the same throughout a simulation run.
- Jobs are released at constant intervals.
- Outliers in completion time are not taken into considerations.
- Transport time of jobs are neglected.
-

4.2.6 Validation and verification

Important factors in simulating processes are the validation and the verification of the model that is used.

Validation

The validity of a simulation model concerns whether the model is a good representation of the real system. The actions that we performed in terms of validation are weekly meetings with the problems owners from the manufacturer. In these meetings we discussed the conceptual models of the assembly line, as well as showing the actual simulation model. These meetings led to the validation of the simulation model, in an iterative way.

Verification

The verification of the simulation model concerns whether the conceptual model is correctly implemented into the simulation software. We performed verification by running several test experiments. In these test experiments we checked whether the simulation output corresponded to the expected output, based on the given input. In addition, we did a lot of manual jumping through events, where we checked if the expected actions for each event were performed correctly.

4.3 Conclusion

We started in Section 4.1 by explaining how we apply the scheduling strategy from Chapter 3 in the simulation model. Also, we provided an overview of the dispatching rules that we tested in our simulation model.

In Section 4.2 we elaborated on the details of the simulation model. We used conceptual models to explain the simulation. In addition, we specified how we used the input data.

5 Solution tests

In Chapter 5, we compare the performance of selected dispatching rules, by using our simulation model. We start by providing the experiment setup in Section 5.1. In Section 5.2 we show the results from the simulation output. Afterwards, Section 5.3 is used to analyze the performance of the scheduling algorithms. We finalize the chapter in Section 5.4 with a conclusion.

5.1 Experiment setup

To find out how the selected dispatching rules perform, we use a total of 4 experiments. In each of the experiments we perform 20 replications, for each of the chosen dispatching rules. In the first experiment, we use the real input data. In the second experiment we use randomly generated data, to investigate whether less uncertainty in completion times has an effect on the performance. The third experiment is used to investigate the effects of less uncertainty in completion times. In experiment 4, we investigate combining dispatching rules.

1) Experiment with real dataset

In this experiment, we test how the chosen dispatching rules perform on the dataset containing the real actual completion times. In the scheduling algorithms there is some randomness involved in the assignment of jobs, because some jobs get the same score. For example, in the FCFS rule, a lot of jobs have release date 0. As a result, there is some luck involved whether the choice was good. Therefore, we perform 20 replications, and in each replication the choice between jobs with the same score is randomized.

2) Experiment with randomly generated dataset

In this experiment we test the scheduling algorithms on randomly generated dataset, which contains different deviations in estimated and actual completion times for each experiment. The details of the input data are given in Section 4.2.3.

3) Experiment with randomly generated dataset with less uncertainty in completion times

In this experiment, we consider less uncertainty in completion times. This can lead to interesting results as well, since the manufacturer is looking to increase the accuracy of their estimated completion times. Also, from a scientific standpoint it is interesting to see how the performance changes when the variance of completion times changes.

We now use different parameters when sampling deviation percentages from actual and estimated completion times from normal distributions. We set the mean equal to 0 and the standard deviation equal to 15, which are considerably better than in the second experiment. Also, we change the upper and lower bounds. The bounds are now set to fall within two standard deviations of the mean, or in other words the values are roughly within the 2,5th and 97,5th percentile. The input parameters we use for sampling the deviation percentages are shown in Table 3.

Table 3: Input parameters for the normal distributions used for sampling deviation percentages.

Stage	Mean	Sigma	Lower bound	Upper bound
1	0	15	-30	30
2	0	15	-30	30

4) Experiment with randomly generated dataset with combination of dispatching rules
In the final experiment we investigate the effects of choosing different scheduling algorithms at different stages. This means that we choose a different scheduling algorithm at the front buffer and the intermediate buffer. We do not consider all combinations, but only combinations of the most promising scheduling algorithms from experiment 1. Therefore, we use the combinations of EDD, MS and FCFS.

5.2 Experiment results

In the following section, we give the results of the simulation experiments. For each experiment, we discuss the experiments in a general way. More detailed results, containing not only averages, but also standard deviations and confidence intervals of each of the KPIs can be found in Appendix G: Detailed output simulation runs.

5.2.1 Experiment 1

In the first experiment, when we test the dispatching rules on the real data, we obtain the following results, shown in Table 4.

Table 4: Averages on KPIs from experiment 1, with colors indicating the relative performance.

KPI	EDD	MS	FCFS	ATC	PT+SL	SPT	Random
Makespan	703,98	667,00	716,81	739,61	764,88	796,88	718,02
Max tardiness	274,86	239,18	300,78	317,77	353,11	590,66	439,36
Avg tardiness	39,60	83,76	41,26	50,43	42,66	41,21	61,56
Percentage late	49,78%	86,12%	40,73%	75,22%	29,91%	26,72%	42,07%
Avg throughput time	105,88	119,78	105,98	119,91	114,32	114,41	107,66
Avg flowtime	244,85	314,75	227,35	279,13	191,20	178,82	231,49
Nr of blocks	0,05	8,35	0,35	1,95	6,65	7	0,35
Nr of starvations	62,00	44,50	64,35	32,00	61,85	48	58,95

From the table, we see that most of the dispatching rules only score well in certain areas. Especially the MS and the SPT score well in certain areas, but poorly in others. The EDD, FCFS and PT+SL perform the best overall. All though, the PT+SL is outperformed by EDD and FCFS in most areas. Except from the random rule, the ATC rule has the worst overall performance. The random rule scores around average for the makespan and percentage of late jobs, but poorly on the maximum and average tardiness.

Apart from the averages over 20 replications, we are also interested in the variance of the KPIs. For example, a KPI can have a good average, but if there is a lot of variance in outcomes for each replication, then the rule is less robust. We only include the most promising rules for further analysis; EDD, FCFS and MS. Also, the random rule is included, to show the effects of using dispatching rules in comparison with choosing randomly. We analyze the most important KPIs, of the most promising rules by using boxplots, which show the minimum and maximum as well as the median, mean and the range between the first and third quartile.

Makespan

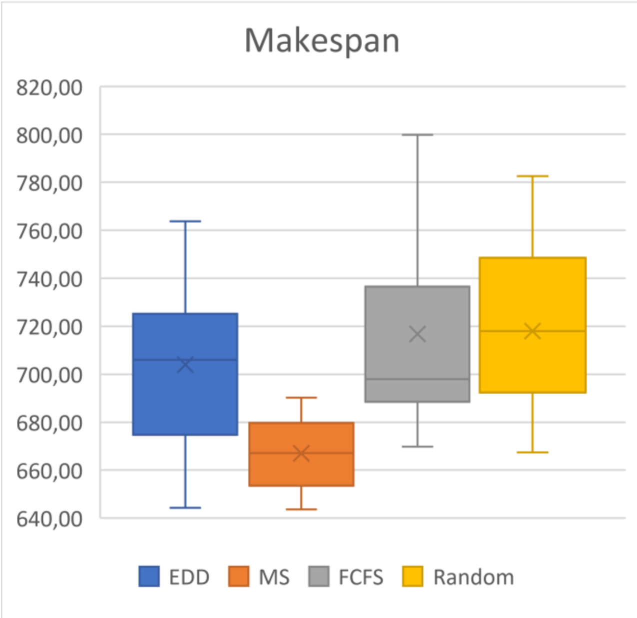

Figure 8: Boxplot of makespan over 20 replications in experiment 1.

From Figure 8 we can conclude that MS rule outperforms the other rules in terms of makespan. In addition, the spread of the makespan for the MS rule is smaller than the other rules, meaning the MS rule is more robust. This is what we expected, since there is more randomness involved in the other rules. Obviously for the random rule we expect more spread in the outcome. Also, in the EDD and FCFS rule there is a lot of randomness involved, due to jobs with equal scores. For example, there are a lot of jobs with the same due date, while there are only a few jobs with the exact same amount of slack.

Maximum tardiness

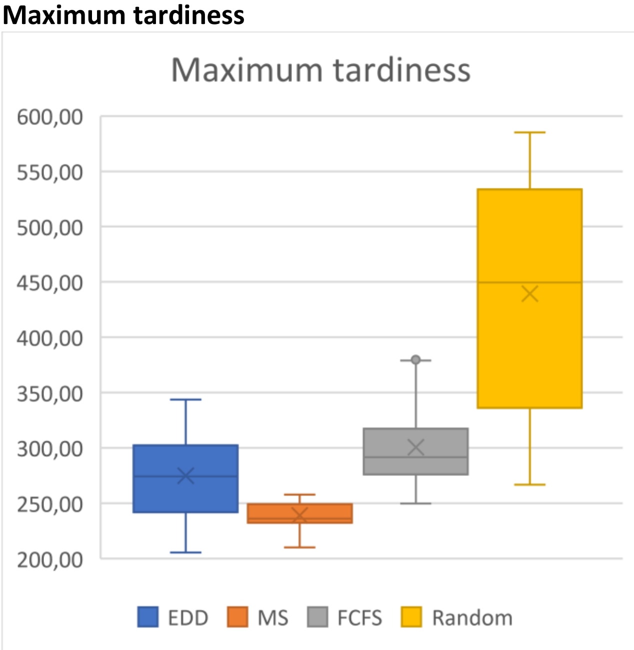

Figure 9: Boxplot of maximum tardiness over 20 replications in experiment 1.

Again, the MS rule outperforms the other rules. Both the average and the spread are smaller for the MS rule. The random rule is clearly outperformed by the others, in terms of maximum tardiness.

Blocking and starvation
The results in Table 4 show that starvation occurs significantly more frequently than blocking. Furthermore, most of the results show combined starving and blocking numbers between 50 and 70. The exception is the ATC rule, which scores significantly lower on blocking and starving, but worse on overall performance. Therefore, we conclude there is no relationship between the number of combined blocks and starvations, and the performance of dispatching rules, based on our results.

5.2.2 Experiment 2

In the second experiment, we used the randomly generated data, where the deviations between the actual and estimated completion time differed for each run. The results are shown in Table 5.

Table 5: Averages on KPIs from experiment 2, with colors indicating the relative performance.

KPI	EDD	MS	FCFS	ATC	PT+SL	SPT	Random
Makespan	784,81	713,67	805,69	792,04	835,84	845,50	811,59
Max tardiness	374,94	316,26	428,44	360,93	443,51	649,45	498,89
Avg tardiness	43,64	82,32	56,22	56,06	46,52	45,60	68,43
Percentage late	47,24%	80,09%	44,44%	70,04%	32,54%	27,11%	46,51%
Avg throughput time	137,88	151,96	131,65	140,19	120,27	117,35	131,82
Avg flowtime	242,42	309,60	240,24	279,95	190,14	178,45	242,79
Nr of blocks	4,80	18,45	3,25	13,65	0,90	0,00	3,10
Nr of starvations	29,80	18,20	42,35	23,45	39,35	41,60	38,90

Due to the use of the randomly generated data, the KPIs in the second experiment are overall slightly higher than in the first experiment. However, the results are very similar to the results of the first experiment, in terms of patterns. Overall, the dispatching rules show the same relative performance as in the first experiment. However, there are a few exceptions. The first one is the SPT rule, which seems the be slightly better in relative performance, than in the first experiment. Secondly, while the ATC is also a bit better than the first experiment, the overall performance is still the worst. Overall, we conclude that the relative performance from the first experiment, shows the same patterns in relative performance as in the second experiment.

5.2.3 Experiment 3

In experiment 3, we used the randomly generated data, this time with smaller deviations in completion times. The results are shown in Table 6.

Table 6: Averages on KPIs from experiment 3, with colors indicating the relative performance.

KPI	EDD	MS	FCFS	ATC	PT+SL	SPT	Random
Makespan	719,83	675,21	744,34	779,32	778,53	772,18	758,31
Max tardiness	278,43	239,71	338,63	354,27	378,62	588,23	486,24
Avg tardiness	43,26	87,09	50,62	58,41	49,18	45,62	70,09
Percentage late	53,36%	91,81%	45,60%	82,67%	33,28%	26,51%	46,68%
Avg throughput time	121,56	147,42	117,45	137,84	113,09	108,32	120,27
Avg flowtime	248,74	319,94	239,57	289,15	194,27	177,90	243,02
Nr of blocks	0,60	15,30	0,75	11,50	0,00	0,10	0,40
Nr of starvations	33,15	13,80	43,70	17,45	40,35	46,55	41,90

The results of experiment 3 are very similar to the results of experiment 2. As expected the KPIs are overall slightly better in the third experiment, because of the lowered degree of stochasticity in deviations between estimated and actual completion time. However, there are two notable exceptions. The first exception is the max tardiness, which has improved a lot for almost all dispatching rules. The improvement in maximum tardiness is expected, since the maximum tardiness is likely to be influenced by outliers in completion times. Because we narrow down the deviations in completion times, the smaller the maximum tardiness becomes. The second exception is the percentage late, which appear to be slightly worse than the second experiment. Overall we conclude that the relative performance has not changed significantly when the completion estimates are closer to the actual times.

5.2.4 Experiment 4

In experiment 4 we investigated some combinations of scheduling algorithms, with the real data. The results are shown in Table 7.

Table 7: Averages on KPIs from experiment 4, with colors indicating the relative performance.

KPI	EDD/MS	EDD/FCFS	MS/EDD	MS/FCFS	FCFS/EDD	FCFS/MS
Avg tardiness	704,16	708,34	668,03	664,92	710,56	708,40
Max tardiness	275,15	276,51	232,79	233,63	289,63	290,51
Percentage late	39,53	39,50	83,49	83,72	41,20	41,23
Makespan	49,61%	49,61%	86,12%	86,12%	40,73%	40,95%
Avg throughput time	105,88	105,87	119,50	119,76	105,97	105,96
Avg flowtime	244,88	244,88	314,51	314,75	227,35	227,33
Nr of blocks	0,00	0,00	8,60	8,80	0,30	0,00
Nr of starvations	63,90	63,40	44,40	44,00	64,90	66,35

In addition to the confidence intervals of the averages, we also provide tables with comparisons to the results of experiment 1, in Appendix G: Detailed output simulation runs. The results from this experiment show that using different dispatching rule for the front buffer and the intermediate buffer, show very little change in compared with using the dispatching rule of the front buffer also at the intermediate buffer. For example, the results of using EDD on both buffers are almost the same as using EDD/MS or EDD/FCFS. While the values differ slightly, the confidence intervals as shown in Appendix G: Detailed output simulation runs, are mostly overlapping, from which we conclude that there is not enough evidence to conclude that a combination of dispatching rules has any effect on the performance.

5.3 Analysis of dispatching rules

EDD

The EDD rule, scores very well from an overall perspective. However, in Figure 8 and Figure 9, we see a relatively high variance in KPIs, meaning the rule is less robust. In comparison with the FCFS rule, which scores similarly in most KPIs, the average tardiness appears to be lower, while the percentage late is a bit higher.

MS

The MS rule outperforms the other rules, in terms of maximum tardiness and the makespan, at the cost of the average tardiness and the percentage late. In addition, both flowtime and throughput time are ~~very~~ high.

FCFS

Similarly to the EDD rule, the FCFS scores well from an overall perspective, making it a good option.

ATC

The ATC rule is by far the worst scoring rule. In general, the more advanced dispatching rules are, the worse the scores on our KPIs are.

PT+SL

This rule performs alright from on overall perspective, but still worse than the EDD and FCFS. However, the PT+SL rule is good for minimizing the percentage of late jobs. In addition, the PT+SL rule scores well on throughput time and flow time.

SPT

As expected, the SPT rule score very well on average tardiness and percentage late. Also average throughput time and average flow time are exceptionally low, but at the cost of a high makespan and an high maximum tardiness.

Random

The random rule scores around average for most KPIs and below average for the maximum tardiness and the average tardiness. The biggest issue with the random rule is the variance of the KPIs, meaning that the KPIs vary a lot per run, due to the randomness.

5.4 Conclusion

Throughout this chapter, we presented the results of our simulation study. We started in Section 5.1 by explaining our experiment setups, for a total of 4 experiments. In experiment 1 we used the real data and in experiment 2 we used the randomly generated data, where the deviations in actual and estimated completion times differ for each experiment. In experiment 3 we used randomly generated data again, but with less variability in deviations, to answer RQ 3B: "*What is the effect of variability in completion times on the KPIs?"* In experiment 4 we look at possible combinations of dispatching rules, with the real dataset.

The results are shown in Section 5.2, answering RQ 3A: *"What are the scores of the scheduling strategies on the KPIs?"* To answer RQ 3B, we observed that changing the variability in completion times does not significantly influence the relative performance of the dispatching rules.

In Section 5.3 We answered RQ 3:" *How do the relevant scheduling strategies perform?"* We concluded that the ATC is the worst rule. The SPT rule scores well in terms average tardiness and percentage late, but performs very poor on the other KPIs. The PT+SL rule is alright in overall performance, but is out performed by the EDD and FCFS rule, which show the best overall performance. The MS rule performs well, in our two most important KPIs, maximum tardiness and makespan, but performs worse than EDD and FCFS in the other KPIs. However, the MS rule is more robust than the EDD and FCFS rule.

We compared using dispatching rules with using a random rule, and conclude that the use of dispatching rules show superior performance over the random rule. Especially the use of dispatching rules shows improvement in the maximum tardiness, and the robustness of the performance.

6 Conclusion

In this final chapter, we conclude on our research. In Section 6.1, we answer our research questions. In Section 6.2 we discuss the limitations of our research and possible further research.

6.1 Conclusion

The goal of this research was to propose a scheduling strategy for the assembly line at the manufacturer. To achieve this goal we divided our research in three main research questions:

What defines a suitable scheduling strategy for the assembly line at the manufacturer?

By answering the first question we found out that there are quantitative and qualitative needs for a scheduling strategy. The quantitative needs are captured in KPIs and are related due dates and efficiency of the assembly process. The qualitative needs are related to the implementation of the scheduling strategy.

What relevant dynamic scheduling strategies are available in literature?

By answering our second main question, we conclude that an event driven policy with complete rescheduling, using dispatching rules, is most suitable for the assembly line at the manufacturer.

How do the relevant scheduling strategies perform?

To answer the last research question we tested several dispatching rules and came to the following conclusions. The results of the simulation show that the ATC rule is not a good option, because it is outperformed by all other rules. Also, the SPT rule did not perform well, especially the high maximum tardiness made the SPT a poor choice, since this is one of our most important KPIs. While the overall performance is alright, the PT+SL is outperformed in most KPIs by the EDD and FCFS, making PT+SL a less interesting option.

Because of its simplicity and overall good performance, the EDD and FCFS are suitable options. In addition the MS is a suitable option. The MS scores poorly in percentages of jobs on time and the average tardiness, but scores well in maximum tardiness and makespan, which we consider as the two most important KPIs.

Furthermore, the MS rule is more robust than the EDD and FCFS rule, meaning that there is less variance in the performance of the MS rule. Therefore, we recommend to implement the MS rule.

We conclude our research by commenting on the objective we stated in Section 1.3.1:

"Propose a dynamic scheduling strategy for the assignment of jobs to workstations in the assembly line at the manufacturer."

In the dynamic scheduling strategy we propose, we use an event driven policy to handle the stochastic release dates and processing times. In this event driven policy, a new schedule is constructed from scratch, each time a specified event occurs. The events that we specify are job completion and job arrivals.

For the construction of a new schedule, we suggest using dispatching rules, because of their fast computation time and simplicity. More specifically, we suggest to implement the MS rule.

6.2 Contribution to practice and science

In this section, we discuss the contribution of our research to both practice and science.

Contribution to practice

We establish the need for a dynamic scheduling strategy, used to cope with the uncertainly in arrival times and completion times. Furthermore, our simulation results provide a list of dispatching rules, with scores on KPIs. Also, we show that using different dispatching rules at the front buffer and the intermediate buffer has no significant benefits. In addition, the results from experiment 3 show that the relative performance of dispatching rules does not change significantly when there is less uncertainty in completion times. Based on the results we provide recommendations on the choice of dispatching rules. The expected gain over scheduling randomly, is mostly in the maximum tardiness and the robustness of performance.

The simulation model that we use during our research contributes to practice as well, since the model can be used for further investigation of scheduling possibilities. In addition, by expanding the simulation model, the manufacturer can investigate other improvements to the assembly process, such as transport capacity, changing the number of parallel workstations at each stage, etcetera.

Contribution to science

Our research introduces the use of dynamic scheduling using dispatching rules in a unique variation of the HFS, namely a two stage assembly line with an intermediate buffer and an individual inventory spot for each parallel workstation at each stage. Also, our research focusses on the presence of highly stochastic processing times, while literature tends to focus on either deterministic processing times or less stochastic processing times.

First results show that simple dispatching rules, focused on due dates, outperform rules focused on utilization or rules focused on multiple objectives. More specifically, the EDD, FCFS, show the best overall performance, while the MS rule is superior in terms of maximum tardiness and makespan.

6.3 Limitations and further research

In this section, we discuss the limitations that our research has and suggest further research to overcome them. The main reason for the limitations are the limited amount of time and the limited amount of available production data.

1) The first limitation is the input data used for the simulation model. During our research we worked with limited data, which has an influence to the validity of the results. Collecting more data on deviations between estimated and actual completion times can be used to verify our results.
2) A simulation model always comes with limitations, because you have to make assumptions and simplifications. Eliminating some of these assumptions and simplifications, as discussed in section 4.2.5, can improve the validity of our research. Another suggestion is to test the scheduling algorithms in the real system, as this is the most accurate way to test the performance.
3) Many more methods on scheduling are available in literature. Testing more methods can result in finding better solutions. A good way to start would be testing more dispatching rules. While more advanced techniques can be an interesting option as well, our recommendation is to first increase the accuracy of estimating completion times, since the more advanced techniques tend to perform worse the more randomness is involved.
4) As discussed in Section 4.1.1, leaving workstations empty on purpose can be an interesting option. Further research has to be done on how this can be used in a beneficial way.
5) The experiments on the effects of variability of completion on the KPIs are limited. More experiments are needed to make solid conclusions.

References

Chen, J., Wang, M., Kong, X., Huang, G., Dai, Q., & Shi, G. (2019). Manufacturing synchronization in a hybrid flowshop with dynamic order arrivals. *Journal of Intelligent Manufacturing*, 2659-2668.

Fernandez-Viagas, V., & Framinan, J. M. (2020). Design of a testbed for hybrid flow shop scheduling with identical machines. *Computers and Industrial Engineering 141*.

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

Jianyu, L., Sun, Z., M, P. P., Bai, Y., Zhang, S., & Li, C. (2020). A robust dynamic scheduling approach based on release time series forecasting for the steelmaking-continuous casting production. *Applied Soft Computing Volume 92*.

Luo, H., Huang, G. Q., Zhang, Y., Dai, Q., & Chen, X. (2009). Two-stage hybrid batching flowshop scheduling with blocking and machine availability constraints using genetic algorithm. *Robotics and Computer-Integrated Manufacturing 25*, 962-971.

Ouelhadj, D., & Petrovic, S. (2009). Survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling*, 417-431.

Peng, K., Pan, Q.-K., Gao, L., Li, X., Das, S., & Zhang, B. (2019). A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm and evolutionary computation*, 92-112.

Pinedo, M. (1995). *Scheduling theory, algorithms and systems.* New Jersey: Prentice Hall.

Pinedo, M. L. (2016). *Scheduling theory, algorithms and systems.* New York: Springer.

Qin, W., Zhang, J., & Song, D. (2018). An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *Journal of intelligent manufacturing*, 891-904.

Shi, L., Guo, G., & Song, X. (2019). Multi-agent based dynamic scheduling optimisation of the sustainable hybrid flow shop in a ubiquitous environment. *International Journal of Production Research*.

Weng, W., Wei, X., & Fujimura, S. (2012). Dynamic routing strategies for JIT production in hybrid flow shops. *Computers and Operations Research*, 3316-3324.

Appendix A: Flowcharts of events

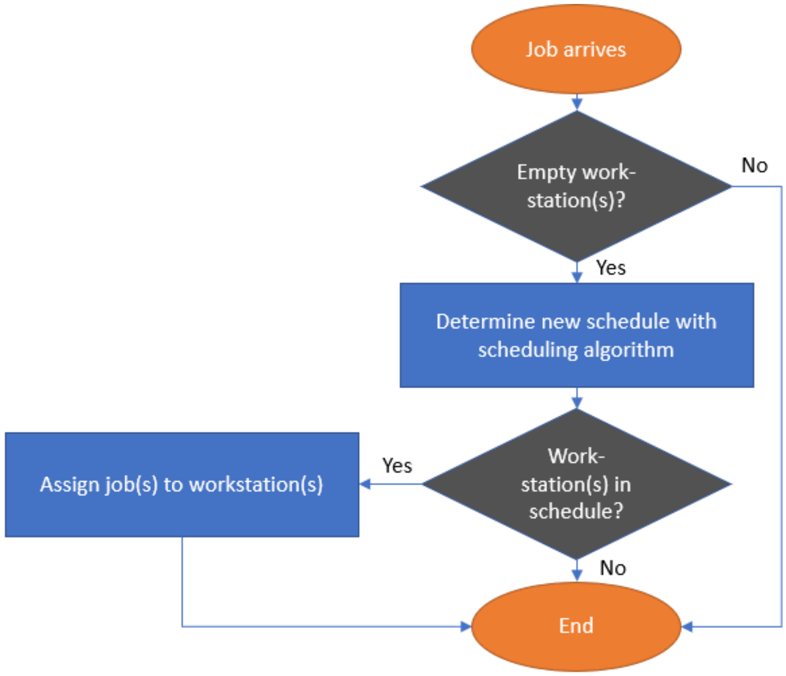

Figure 10: Flowchart of the general dynamic scheduling strategy in case of a job arrival event.

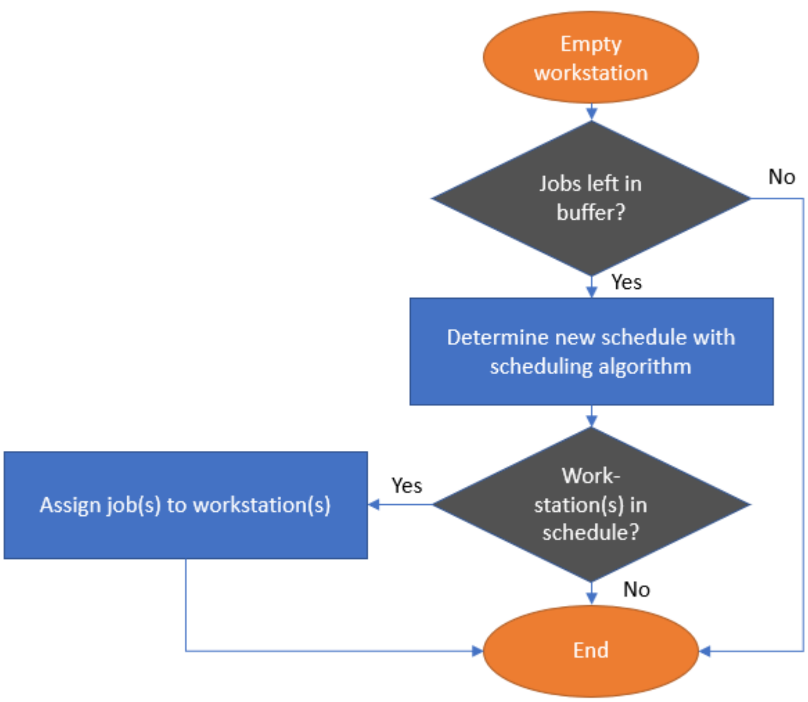

Figure 11: Flowchart of the general dynamic scheduling strategy in case of a an empty workstation event.

Appendix B: Conceptual models for simulation model

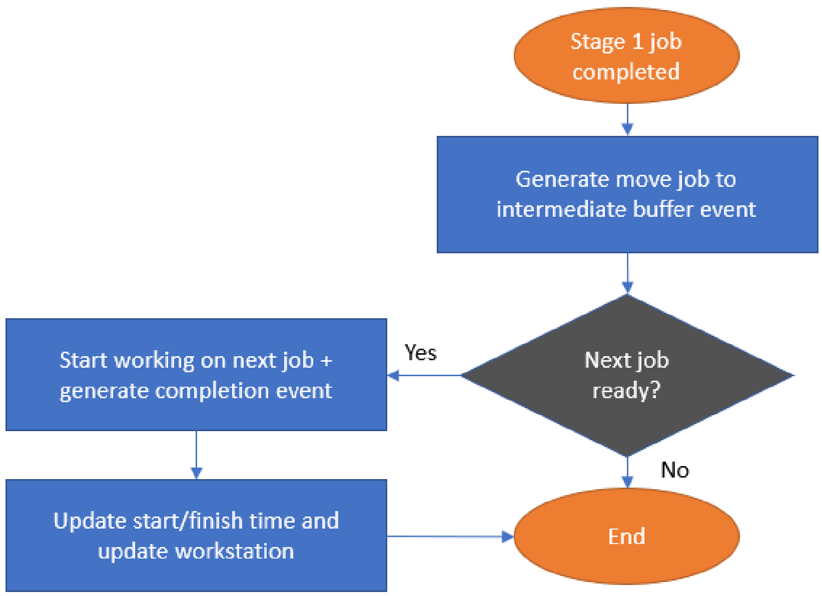

Figure 12: Flowchart of stage 1 job completed event

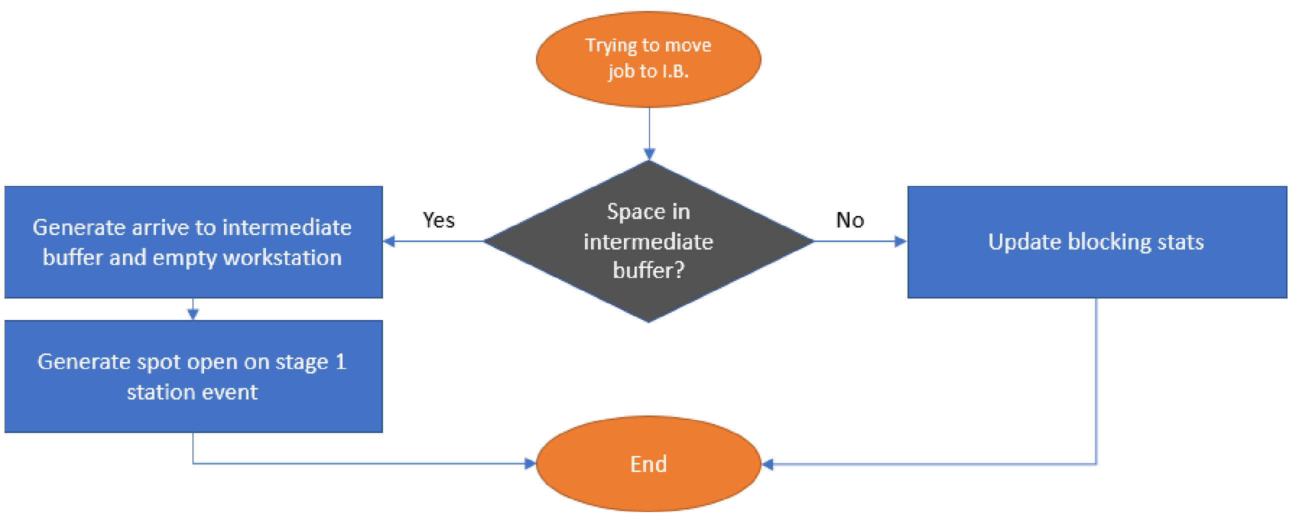

Figure 13: Flowchart of trying to move a job to the intermediate buffer event.

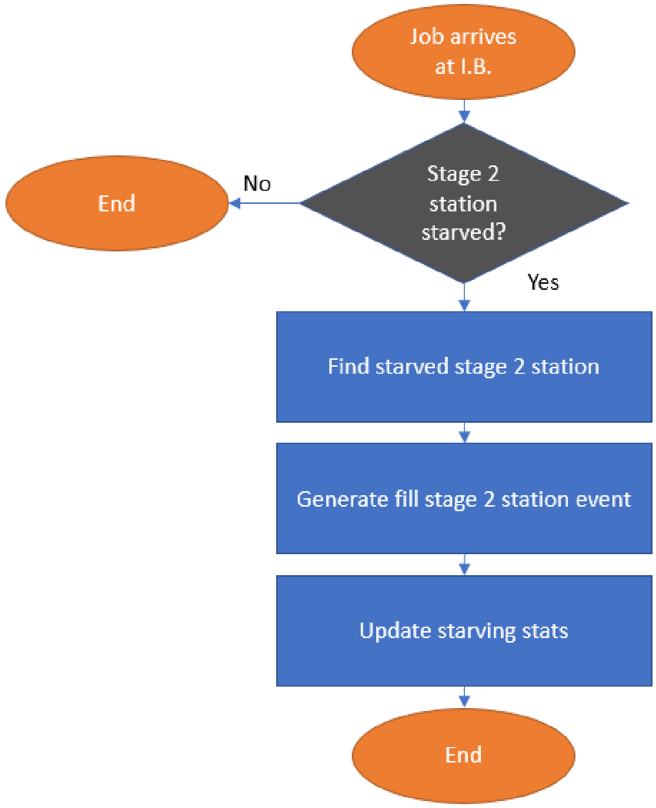

Figure 14: Flowchart of a job arrival at the intermediate buffer event.

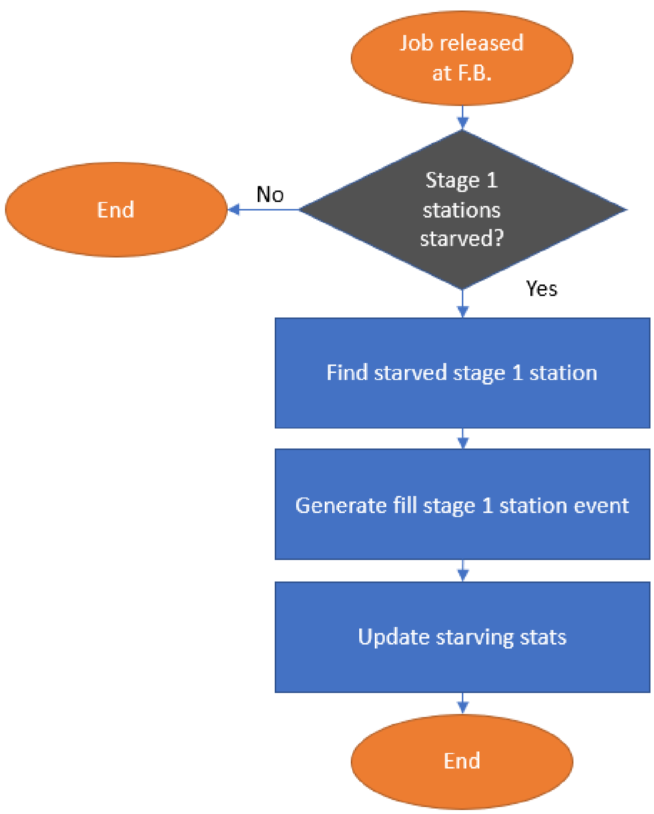

Figure 15: Flowchart of job release at the front buffer event.

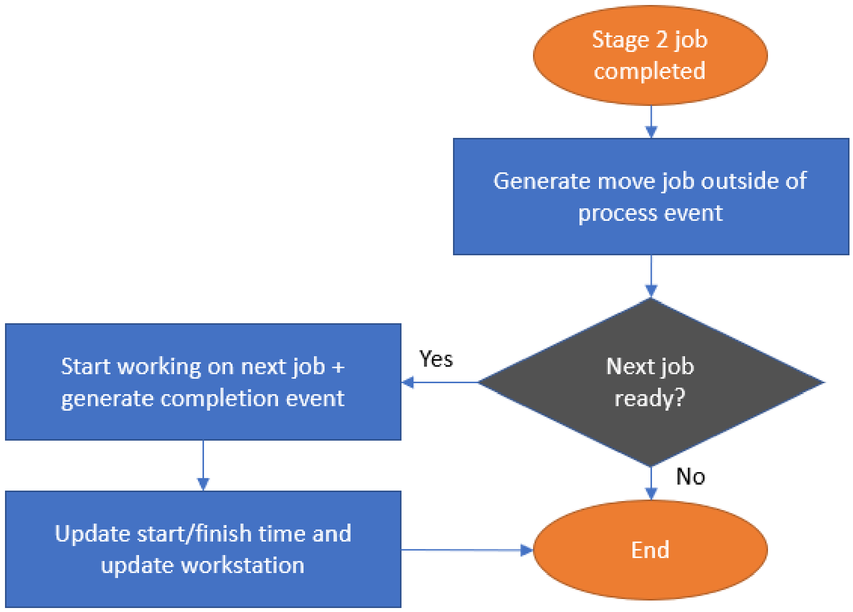

Figure 16: Flowchart of stage 2 job completed event.

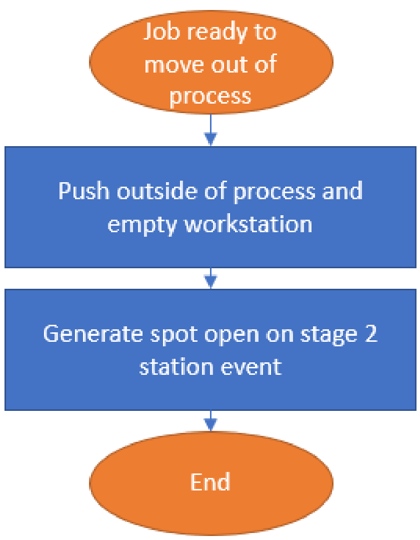

Figure 17: Flowchart of moving a stage 2 job out of the process event.

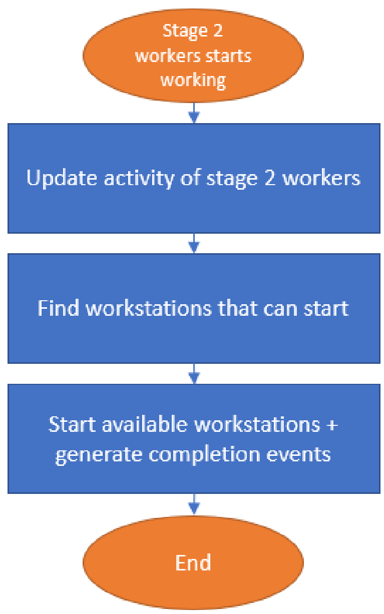

Figure 18: Flowchart of stage 2 workers being released event.

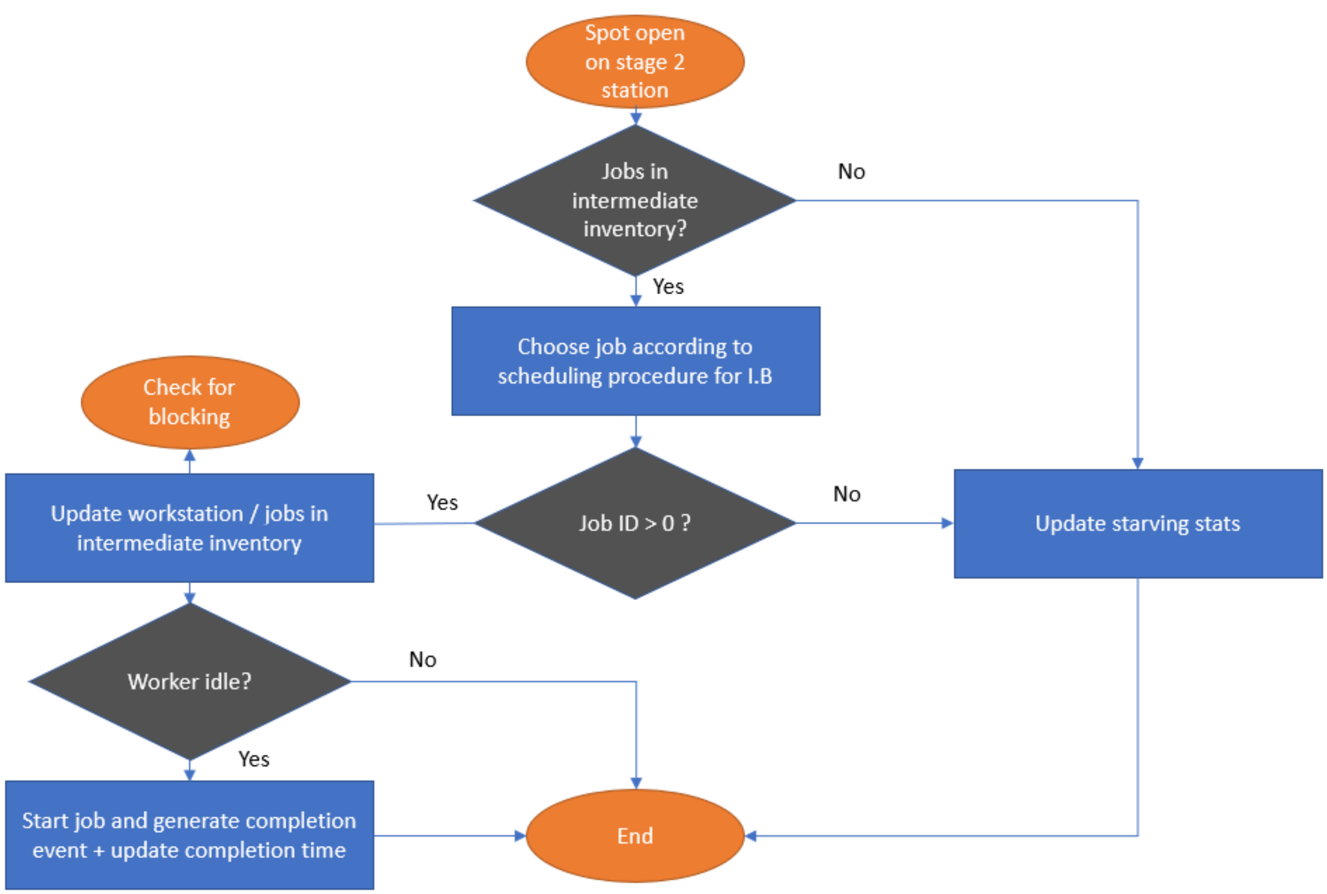

Figure 19: Flowchart of a stage 2 workstation that is emptied event.

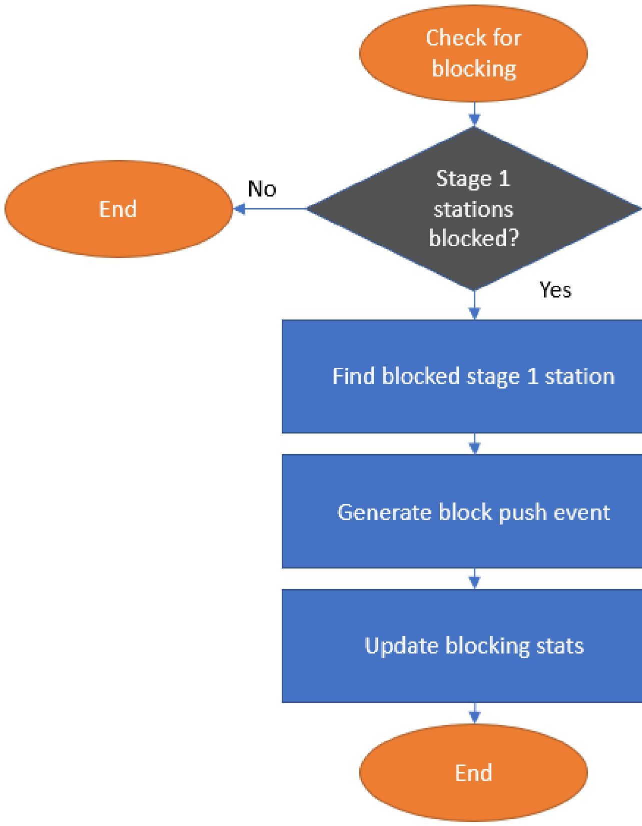

Figure 20: Flowchart of checking for blocking event.

Appendix C: Algorithm used for scheduling in simulation model

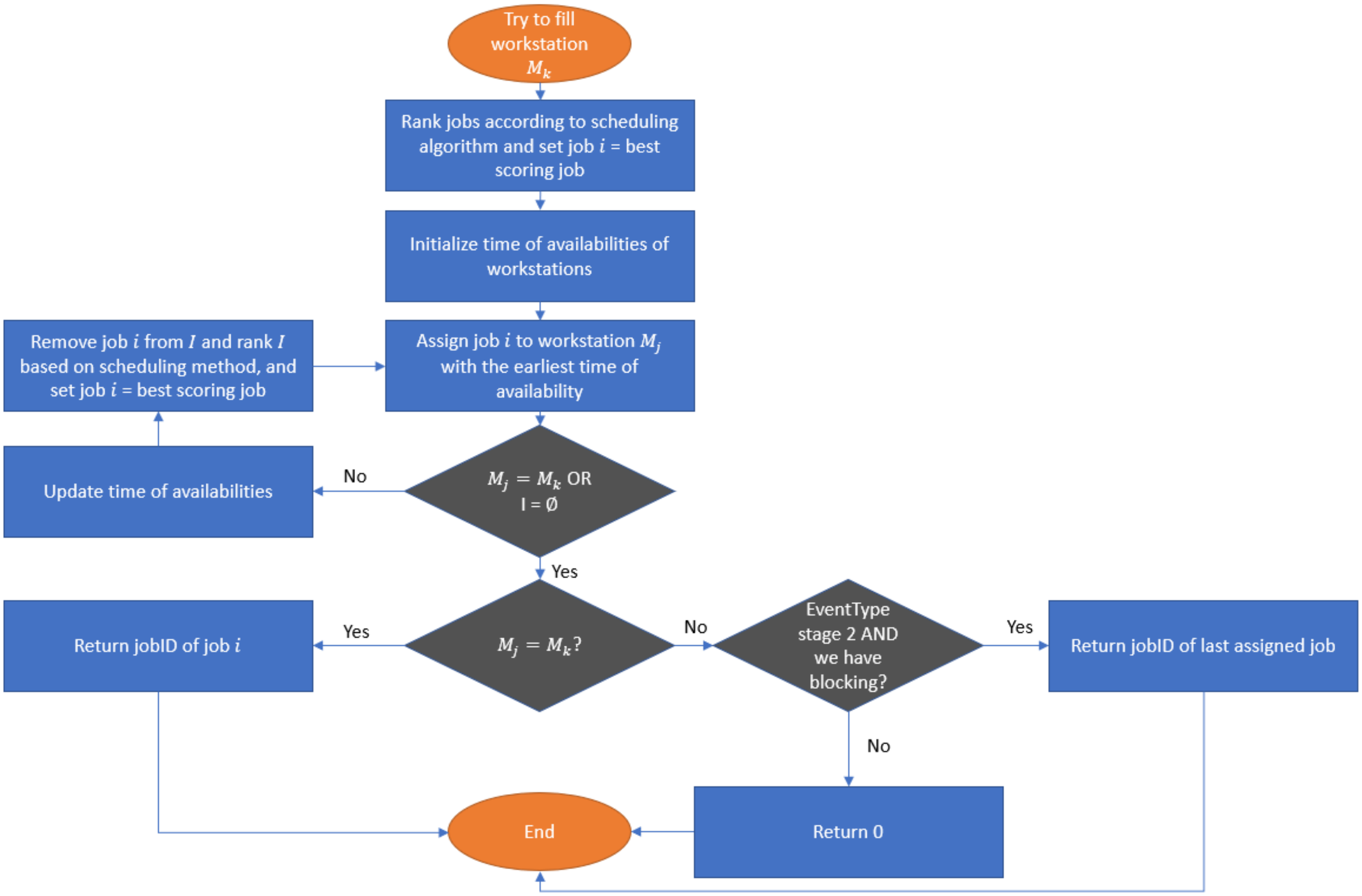

Figure 21: Algorithm used for scheduling in the simulation model, to save computation time.

Appendix D: Input data

Table 8: Input data with real estimated and actual completion times.

Beam ID	Due date	Release date	Estimated stage 1	Actual stage 1	Estimated stage 2	Actual stage 2
1	240	0	12,50	3,87	36,27	20,70
2	240	0	10,00	3,25	42,20	31,55
3	240	0	10,00	4,48	42,20	30,20
4	240	0	27,50	18,78	39,07	50,70
5	240	0	12,50	4,14	19,47	21,82
6	240	0	12,50	10,40	21,82	21,78
7	240	0	10,00	5,49	42,20	24,47
8	240	0	22,50	16,79	32,27	32,90
9	240	0	12,50	11,48	19,20	32,10
10	240	0	10,00	4,36	45,50	36,65
11	180	0	27,50	25,54	105,72	66,68
12	240	0	22,50	20,98	43,50	43,12
13	180	0	27,50	20,29	100,37	74,72
14	240	0	30,00	26,46	120,20	98,87
15	240	0	10,00	6,36	17,17	28,40
16	240	0	12,50	4,93	36,27	38,07
17	240	0	15,00	11,12	28,02	29,07
18	180	0	10,00	10,79	42,40	32,23
19	240	0	10,00	2,41	42,20	28,87
20	180	0	22,50	17,13	37,73	46,82
21	240	0	15,00	12,95	28,02	32,42
22	240	0	10,00	2,86	42,20	29,62
23	180	0	10,00	8,07	37,12	34,23
24	240	0	12,50	22,28	20,68	22,32

25	240	0	10,00	3,56	42,20	36,18
26	240	0	10,00	7,24	17,17	27,47
27	180	0	27,50	16,26	57,62	56,93
28	240	0	20,00	25,06	38,53	58,53
29	420	60	20,00	37,87	60,08	55,20
30	420	168	67,50	82,08	251,42	175,30
31	420	12	10,00	4,54	21,55	42,40
32	420	120	5,00	6,88	14,00	11,18
33	420	90	10,00	1,98	21,55	35,43
34	420	174	15,00	22,18	108,63	74,23
35	360	138	22,50	54,46	157,13	147,37
36	420	36	10,00	17,29	28,00	32,45
37	420	24	10,00	4,83	21,55	27,70
38	420	144	5,00	0,00	14,00	37,05
39	420	102	5,00	7,22	14,00	18,82
40	420	54	67,50	123,14	254,93	216,77
41	420	48	17,50	26,33	134,35	240,45
42	420	96	5,00	7,71	14,00	28,15
43	420	180	30,00	27,41	217,28	141,38
44	420	78	17,50	19,18	55,12	55,77
45	420	108	27,50	20,52	64,97	59,32
46	420	156	5,00	7,50	14,00	15,80
47	480	126	15,00	26,53	83,45	66,80
48	420	72	5,00	6,06	14,00	37,05
49	420	84	10,00	5,28	21,55	55,00
50	420	6	30,00	28,79	72,18	28,45
51	420	150	20,00	44,02	143,25	131,67
52	360	162	22,50	36,29	157,13	104,10

53	420	114	15,00	21,16	108,63	88,23
54	420	30	5,00	5,85	14,00	17,45
55	480	18	10,00	16,38	71,60	61,87
56	420	42	20,00	34,28	127,32	122,37
57	420	132	5,00	6,01	14,00	32,38
58	420	66	10,00	7,14	21,55	28,40
59	240	0	10,00	6,36	17,17	28,40
60	180	0	15,00	22,18	108,63	74,23
61	240	0	10,00	5,49	42,20	24,47
62	180	0	67,50	123,14	254,93	216,77
63	180	0	5,00	6,01	14,00	32,38
64	180	0	5,00	6,06	14,00	37,05
65	240	0	10,00	3,56	42,20	36,18
66	180	0	30,00	27,41	217,28	141,38
67	180	0	10,00	8,07	37,12	34,23
68	180	0	27,50	16,26	57,62	56,93
69	240	0	10,00	4,48	42,20	30,20
70	180	0	10,00	7,14	21,55	28,40
71	240	0	10,00	2,86	42,20	29,62
72	180	0	27,50	25,54	105,72	66,68
73	240	0	15,00	12,95	28,02	32,42
74	180	0	10,00	17,29	28,00	32,45
75	180	0	5,00	5,85	14,00	17,45
76	180	0	30,00	28,79	72,18	28,45
77	180	0	5,00	0,00	14,00	37,05
78	240	0	12,50	10,40	21,82	21,78
79	240	0	20,00	25,06	38,53	58,53
80	180	0	5,00	7,22	14,00	18,82

81	180	0	22,50	36,29	157,13	104,10
82	180	0	10,00	4,83	21,55	27,70
83	180	0	10,00	5,28	21,55	55,00
84	240	0	12,50	4,14	19,47	21,82
85	180	0	20,00	37,87	60,08	55,20
86	180	0	67,50	82,08	251,42	175,30
87	240	0	22,50	20,98	43,50	43,12
88	420	240	20,00	44,02	143,25	131,67
89	420	246	22,50	54,46	157,13	147,37
90	480	192	15,00	11,12	28,02	29,07
91	420	234	10,00	4,54	21,55	42,40
92	420	324	15,00	21,16	108,63	88,23
93	420	342	22,50	17,13	37,73	46,82
94	420	216	5,00	7,50	14,00	15,80
95	480	318	12,50	11,48	19,20	32,10
96	420	348	5,00	7,71	14,00	28,15
97	480	300	10,00	7,24	17,17	27,47
98	480	204	10,00	4,36	45,50	36,65
99	420	330	17,50	19,18	55,12	55,77
100	480	264	12,50	22,28	20,68	22,32
101	480	258	10,00	3,25	42,20	31,55
102	420	282	27,50	20,29	100,37	74,72
103	480	186	22,50	16,79	32,27	32,90
104	420	276	10,00	10,79	42,40	32,23
105	420	252	17,50	26,33	134,35	240,45
106	420	288	20,00	34,28	127,32	122,37
107	420	270	27,50	20,52	64,97	59,32
108	420	210	5,00	6,88	14,00	11,18

109	480	312	27,50	18,78	39,07	50,70
110	480	354	30,00	26,46	120,20	98,87
111	480	222	12,50	3,87	36,27	20,70
112	480	228	10,00	16,38	71,60	61,87
113	480	294	15,00	26,53	83,45	66,80
114	480	306	10,00	2,41	42,20	28,87
115	420	198	10,00	1,98	21,55	35,43
116	480	336	12,50	4,93	36,27	38,07

Appendix E: Choosing distributions and parameters for input data

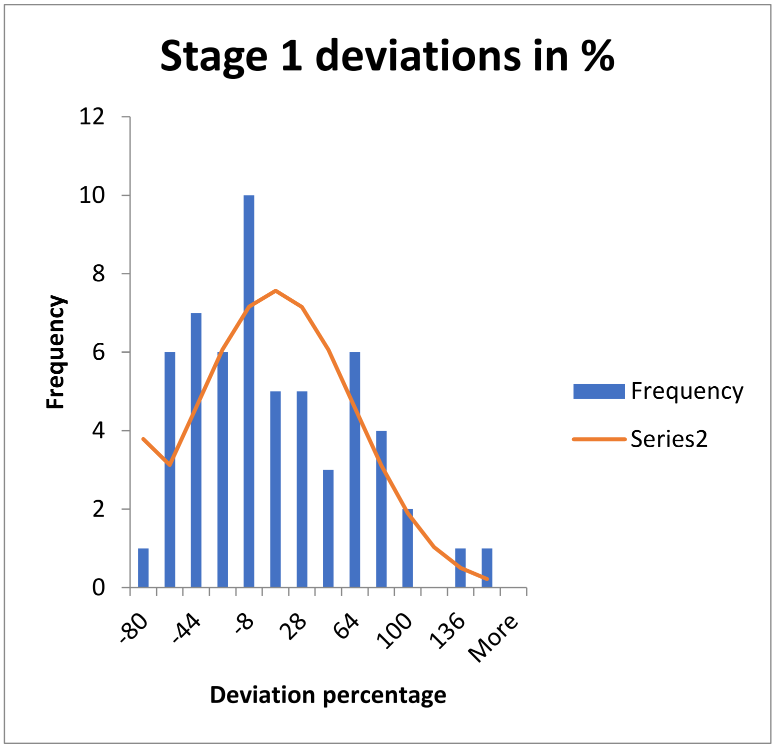

Figure 22: Histogram of deviation percentages of stage 1 jobs.

Table 9: Descriptive statistics of deviation percentages of stage 1 jobs.

Stage 1	
Mean	0,92
Standard Error	7,13
Median	-8,64
Mode	#N/A
Standard Deviation	53,85
Sample Variance	2900,20
Kurtosis	-0,45
Skewness	0,53
Range	222,20
Minimum	-80,17
Maximum	142,04
Sum	52,33
Count	57

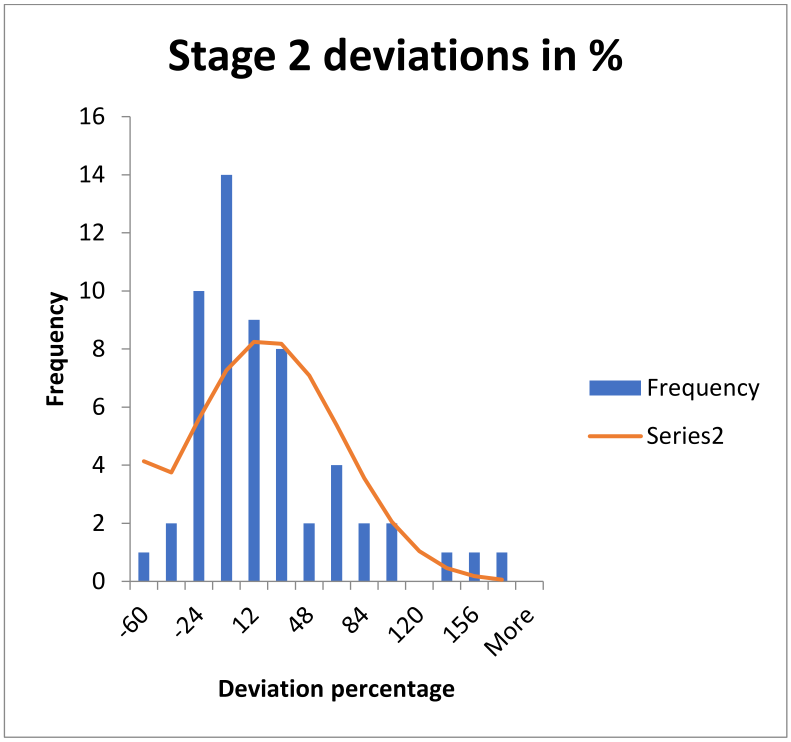

Figure 23: Histogram of deviation percentages of stage 2 jobs in blue bars

Table 10: Descriptive statistics of deviation percentages of stage 2 jobs.

Stage 2	
Mean	10,96
Standard Error	6,45
Median	-1,19
Mode	#N/A
Standard Deviation	48,72
Sample Variance	2373,42
Kurtosis	1,99
Skewness	1,45
Range	225,23
Minimum	-60,59
Maximum	164,64
Sum	624,85
Count	57

Distribution and parameters

Both histograms from stage 1 and stage 2 deviations from actual and estimated completion times show some similarity to a normal distribution, which is plotted, for the corresponding mean and standard deviation, in orange. However, both stages show a trend where most deviations are shifted to the left of the mean, and outliers only occur to the right of the mean.

In the first attempt to see if these deviations represent the original data, we sampled the deviations for stage 1 and stage 2 from normal distributions with their corresponding mean and standard deviations as shown in Table 9 and Table 10. The simulation results, in comparison with the original data, show significantly longer worse KPIs. Therefore, in the second attempt, to compensate for the trends of a large portion of deviations being shifted to left of the mean and outliers being shifted to the right, we now use the medians as parameters for the means of the distributions. In addition, we introduced bounds for sampling, that prevent enormous outliers in the deviation of actual and estimated completion times. The second attempt shows a more accurate representation of the original dataset.

Appendix F: Parametric analysis ATC rule

In order to find the correct scaling parameter for the ATC dispatching rule we perform a parametric analysis. We run the simulation for different values of the scaling parameter, over a total of 5 replications per run. The results are shown in Table 11.

Table 11: Parametric analysis of scaling parameter in ATC dispatching rule.

Parametric analysis ATC	**0,1**	**0,2**	**0,3**	**0,4**	**0,5**	**0,6**	**0,7**	**0,8**	**0,9**	**1,0**
Avg tardiness	60,20	58,63	56,73	53,87	57,34	51,59	50,59	40,01	24,96	27,24
Max tardiness	334,10	358,95	362,44	311,62	336,75	294,99	321,13	365,55	595,94	579,72
Percentage late	84,83%	84,48%	81,55%	80,00%	79,48%	74,31%	71,90%	60,17%	23,45%	20,34%
Makespan	754,10	778,95	782,44	731,62	759,72	715,76	741,78	785,55	786,23	784,89
Avg throughput time	117,37	117,27	119,44	122,21	124,35	119,73	120,36	117,99	104,62	112,38
Avg flowtime	290,72	289,68	287,40	284,14	287,65	280,21	278,27	261,24	200,03	188,08

Since we are looking for overall balance between average tardiness, maximum tardiness, percentage late and the makespan, the values of 0,6 and 0,7 fit the best. We make the final choice of choosing 0,6 because of the better score on max tardiness and makespan.

As expected, the ATC rule start behaving more like the SPT rule when having a larger K. The values of the KPIs at a value of 1,0 is almost equal to those of the SPT rule.

Appendix G: Detailed output simulation runs

Experiment 1

EDD

Table 12: Simulation output for EDD rule in experiment 1.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	703,98	33,05	688,51	719,45
Max tardiness	274,86	36,04	257,99	291,72
Avg tardiness	39,60	5,69	36,94	42,27
Percentage late	49,78%	4,75%	47,56%	52,01%
Avg throughput time	105,88	3,58	104,21	107,56
Avg flowtime	244,85	7,67	241,26	248,44
Nr of blocks	0,05	0,22	-0,05	0,15
Nr of starvations	62,00	10,67	57,01	66,99

MS

Table 13: Simulation output for MS rule in experiment 1.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	667,00	15,26	659,86	674,14
Max tardiness	239,18	12,64	233,27	245,10
Avg tardiness	83,76	1,94	82,86	84,67
Percentage late	86,12%	0,39%	85,94%	86,30%
Avg throughput time	119,78	1,54	119,06	120,50
Avg flowtime	314,75	1,97	313,82	315,67
Nr of blocks	8,35	2,21	7,32	9,38
Nr of starvations	44,50	3,22	42,99	46,01

FCFS

Table 14: Simulation output for FCFS rule in experiment 1.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	716,81	38,82	698,65	734,98
Max tardiness	300,78	34,95	284,42	317,13
Avg tardiness	41,26	1,51	40,55	41,96
Percentage late	40,73%	3,01%	39,32%	42,14%
Avg throughput time	105,98	3,85	104,17	107,78
Avg flowtime	227,35	3,80	225,57	229,13
Nr of blocks	0,35	1,57	-0,38	1,08
Nr of starvations	64,35	10,05	59,64	69,06

ATC

Table 15: Simulation output for ATC rule in experiment 1.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	739,61	46,55	717,82	761,39
Max tardiness	317,77	49,27	294,72	340,83
Avg tardiness	50,43	1,94	49,53	51,34
Percentage late	75,22%	1,89%	74,33%	76,10%
Avg throughput time	119,91	1,80	119,07	120,75
Avg flowtime	279,13	2,31	278,05	280,22
Nr of blocks	1,95	1,90	1,06	2,84
Nr of starvations	32,00	8,91	27,83	36,17

PT+SL

Table 16: Simulation output for PT+SL rule in experiment 1.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	764,88	30,80	750,46	779,29
Max tardiness	353,11	25,52	341,16	365,05
Avg tardiness	42,66	1,18	42,11	43,21
Percentage late	29,91%	1,01%	29,44%	30,39%
Avg throughput time	114,32	0,64	114,03	114,62
Avg flowtime	191,20	1,56	190,47	191,93
Nr of blocks	6,65	1,57	5,92	7,38
Nr of starvations	61,85	8,76	57,75	65,95

SPT; we use only 1 run in the first experiment, because the output of multiple runs will be the same.

Table 17: Simulation output for SPT rule in experiment 1.

KPI	**Mean**
Makespan	796,88
Max tardiness	590,66
Avg tardiness	41,21
Percentage late	26,72%
Avg throughput time	114,41
Avg flowtime	178,82
Nr of blocks	7
Nr of starvations	48

Random

Table 18: Simulation output for random rule in experiment 1.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	718,02	33,66	702,27	733,78
Max tardiness	439,36	103,50	390,92	487,80
Avg tardiness	61,56	17,09	53,56	69,56
Percentage late	42,07%	3,85%	40,27%	43,87%
Avg throughput time	107,66	3,30	106,12	109,21
Avg flowtime	231,49	12,35	225,71	237,26
Nr of blocks	0,35	1,57	-0,38	1,08
Nr of starvations	58,95	7,51	55,44	62,46

Experiment 2

EDD

Table 19: Simulation output for EDD rule in experiment 2.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	784,81	93,89	740,87	828,76
Max tardiness	374,94	106,49	325,10	424,78
Avg tardiness	43,64	12,61	37,74	49,54
Percentage late	47,24%	7,01%	43,96%	50,52%
Avg throughput time	137,88	15,44	130,66	145,11
Avg flowtime	242,42	17,82	234,08	250,76
Nr of blocks	4,80	5,17	2,38	7,22
Nr of starvations	29,80	13,58	23,45	36,15

MS

Table 20: Simulation output for MS rule in experiment 2.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	713,67	63,15	684,11	743,22
Max tardiness	316,26	71,97	282,58	349,95
Avg tardiness	82,32	23,00	71,56	93,08
Percentage late	80,09%	8,89%	75,93%	84,25%
Avg throughput time	151,96	14,15	145,34	158,59
Avg flowtime	309,60	26,67	297,12	322,08
Nr of blocks	18,45	7,16	15,10	21,80
Nr of starvations	18,20	5,74	15,52	20,88

FCFS

Table 21: Simulation output for FCFS rule in experiment 2.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	805,69	82,69	766,99	844,39
Max tardiness	428,44	99,65	381,80	475,08
Avg tardiness	56,22	15,28	49,07	63,38
Percentage late	44,44%	7,96%	40,72%	48,16%
Avg throughput time	131,65	13,41	125,38	137,93
Avg flowtime	240,24	22,37	229,77	250,71
Nr of blocks	3,25	5,88	0,50	6,00
Nr of starvations	42,35	12,44	36,53	48,17

ATC

Table 22: Simulation output for ATC rule in experiment 2.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	792,04	104,05	743,35	840,74
Max tardiness	360,93	109,35	309,75	412,10
Avg tardiness	56,06	14,37	49,34	62,79
Percentage late	70,04%	8,40%	66,11%	73,97%
Avg throughput time	140,19	13,59	133,83	146,55
Avg flowtime	279,95	17,27	271,87	288,03
Nr of blocks	13,65	6,56	10,58	16,72
Nr of starvations	23,45	12,56	17,57	29,33

PT+SL

Table 23: Simulation output for PT+SL rule in experiment 2.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	835,84	92,47	792,56	879,12
Max tardiness	443,51	96,72	398,24	488,77
Avg tardiness	46,52	8,67	42,46	50,58
Percentage late	32,54%	2,37%	31,43%	33,65%
Avg throughput time	120,27	7,95	116,55	123,99
Avg flowtime	190,14	11,00	184,99	195,29
Nr of blocks	0,90	4,02	-0,98	2,78
Nr of starvations	39,35	10,94	34,23	44,47

SPT

Table 24: Simulation output for SPT rule in experiment 2.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	845,50	91,38	802,73	888,27
Max tardiness	649,45	101,00	602,18	696,72
Avg tardiness	45,60	6,09	42,75	48,45
Percentage late	27,11%	2,59%	25,90%	28,32%
Avg throughput time	117,35	8,95	113,16	121,54
Avg flowtime	178,45	7,87	174,77	182,14
Nr of blocks	0,00	0,00	0,00	0,00
Nr of starvations	41,60	15,13	34,52	48,68

Random

Table 25: Simulation output for random rule in experiment 2.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	811,59	92,18	768,45	854,73
Max tardiness	498,89	124,89	440,44	557,33
Avg tardiness	68,43	16,13	60,88	75,97
Percentage late	46,51%	4,60%	44,36%	48,66%
Avg throughput time	131,82	14,13	125,21	138,44
Avg flowtime	242,79	17,73	234,49	251,09
Nr of blocks	3,10	5,00	0,76	5,44
Nr of starvations	38,90	15,36	31,71	46,09

Experiment 3

EDD

Table 26: Simulation output for EDD in experiment 3.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	719,83	48,05	697,35	742,32
Max tardiness	278,43	38,68	260,33	296,54
Avg tardiness	43,26	5,51	40,68	45,83
Percentage late	53,36%	4,15%	51,42%	55,30%
Avg throughput time	121,56	5,80	118,84	124,27
Avg flowtime	248,74	8,04	244,98	252,50
Nr of blocks	0,60	1,57	-0,13	1,33
Nr of starvations	33,15	8,44	29,20	37,10

MS

Table 27: Simulation output for MS in experiment 3.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	675,21	21,67	665,07	685,35
Max tardiness	239,71	41,92	220,09	259,33
Avg tardiness	87,09	8,36	83,18	91,01
Percentage late	91,81%	2,41%	90,68%	92,94%
Avg throughput time	147,42	5,73	144,74	150,10
Avg flowtime	319,94	8,76	315,84	324,04
Nr of blocks	15,30	4,47	13,21	17,39
Nr of starvations	13,80	2,21	12,76	14,84

FCFS

Table 28: Simulation output for FCFS in experiment 3.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	744,34	54,88	718,66	770,03
Max tardiness	338,63	54,23	313,25	364,01
Avg tardiness	50,62	6,03	47,80	53,45
Percentage late	45,60%	4,71%	43,40%	47,81%
Avg throughput time	117,45	6,06	114,61	120,28
Avg flowtime	239,57	8,89	235,41	243,73
Nr of blocks	0,75	3,35	-0,82	2,32
Nr of starvations	43,70	12,57	37,82	49,58

ATC

Table 29: Simulation output for ATC in experiment 3.

KPI	**Mean**	**Sigma**	**L.B C.I (95%)**	**U.B C.I (95%)**
Makespan	779,32	69,04	747,01	811,63
Max tardiness	354,27	73,88	319,69	388,85
Avg tardiness	58,41	4,97	56,09	60,74
Percentage late	82,67%	2,50%	81,50%	83,84%
Avg throughput time	137,84	5,06	135,47	140,20
Avg flowtime	289,15	5,39	286,62	291,67
Nr of blocks	11,50	3,41	9,90	13,10
Nr of starvations	17,45	9,42	13,04	21,86

PT+SL

Table 30: Simulation output for PT+SL in experiment 3.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	778,53	37,84	760,82	796,24
Max tardiness	378,62	32,61	363,36	393,89
Avg tardiness	49,18	2,81	47,86	50,49
Percentage late	33,28%	1,44%	32,60%	33,95%
Avg throughput time	113,09	2,64	111,85	114,32
Avg flowtime	194,27	3,94	192,43	196,11
Nr of blocks	0,00	0,00	0,00	0,00
Nr of starvations	40,35	6,23	37,44	43,26

SPT

Table 31: Simulation output for SPT in experiment 3.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	772,18	28,34	758,92	785,44
Max tardiness	588,23	31,82	573,34	603,12
Avg tardiness	45,62	2,23	44,58	46,66
Percentage late	26,51%	1,34%	25,88%	27,13%
Avg throughput time	108,32	1,86	107,45	109,19
Avg flowtime	177,90	2,54	176,71	179,09
Nr of blocks	0,10	0,45	-0,11	0,31
Nr of starvations	46,55	5,40	44,02	49,08

Random

Table 32: Simulation output for random rule in experiment 3.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	758,31	59,38	730,52	786,10
Max tardiness	486,24	111,81	433,91	538,57
Avg tardiness	70,09	16,25	62,49	77,70
Percentage late	46,68%	4,35%	44,64%	48,72%
Avg throughput time	120,27	8,29	116,39	124,14
Avg flowtime	243,02	10,15	238,27	247,77
Nr of blocks	0,40	0,99	-0,07	0,87
Nr of starvations	41,90	9,28	37,55	46,25

Experiment 4

EDD/MS

Table 33: Simulation output for EDD/MS in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	704,16	32,55	688,92	719,39
Max tardiness	275,15	36,61	258,02	292,29
Avg tardiness	39,53	5,58	36,92	42,14
Percentage late	49,61%	4,62%	47,45%	51,77%
Avg throughput time	105,88	3,63	104,18	107,58
Avg flowtime	244,88	7,80	241,23	248,53
Nr of blocks	0,00	0,00	0,00	0,00
Nr of starvations	63,90	7,68	60,31	67,49

EDD/FCFS

Table 34: Simulation output for EDD/FCFS in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	708,34	34,69	692,11	724,58
Max tardiness	276,51	37,85	258,80	294,23
Avg tardiness	39,50	5,52	36,92	42,09
Percentage late	49,61%	4,63%	47,44%	51,78%
Avg throughput time	105,87	3,57	104,20	107,54
Avg flowtime	244,88	7,76	241,25	248,51
Nr of blocks	0,00	0,00	0,00	0,00
Nr of starvations	63,40	8,04	59,64	67,16

MS/EDD

Table 35: Simulation output for MS/EDD in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	668,03	13,91	661,52	674,54
Max tardiness	232,79	12,97	226,71	238,86
Avg tardiness	83,49	1,87	82,62	84,37
Percentage late	86,12%	0,39%	85,94%	86,30%
Avg throughput time	119,50	1,47	118,81	120,19
Avg flowtime	314,51	1,92	313,61	315,41
Nr of blocks	8,60	2,39	7,48	9,72
Nr of starvations	44,40	3,76	42,64	46,16

MS/FCFS

Table 36: Simulation output for MS/FCFS in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	664,92	15,30	657,76	672,08
Max tardiness	233,63	14,53	226,83	240,43
Avg tardiness	83,72	1,97	82,80	84,64
Percentage late	86,12%	0,39%	85,94%	86,30%
Avg throughput time	119,76	1,53	119,05	120,48
Avg flowtime	314,75	2,01	313,81	315,69
Nr of blocks	8,80	1,51	8,09	9,51
Nr of starvations	44,00	2,10	43,02	44,98

FCFS/EDD

Table 37: Simulation output for FCFS/EDD in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	710,56	34,04	694,63	726,49
Max tardiness	289,63	27,84	276,60	302,66
Avg tardiness	41,20	1,49	40,50	41,90
Percentage late	40,73%	3,16%	39,25%	42,21%
Avg throughput time	105,97	3,76	104,21	107,73
Avg flowtime	227,35	3,74	225,59	229,10
Nr of blocks	0,30	1,34	-0,33	0,93
Nr of starvations	64,90	9,56	60,42	69,38

FCFS/MS

Table 38: Simulation output for FCFS/MS in experiment 4.

KPI	Mean	Sigma	L.B C.I (95%)	U.B C.I (95%)
Makespan	708,40	29,53	694,58	722,22
Max tardiness	290,51	28,52	277,16	303,86
Avg tardiness	41,23	1,56	40,50	41,96
Percentage late	40,95%	3,08%	39,51%	42,39%
Avg throughput time	105,96	3,81	104,18	107,74
Avg flowtime	227,33	3,77	225,57	229,09
Nr of blocks	0,00	0,00	0,00	0,00
Nr of starvations	66,35	8,13	62,54	70,16