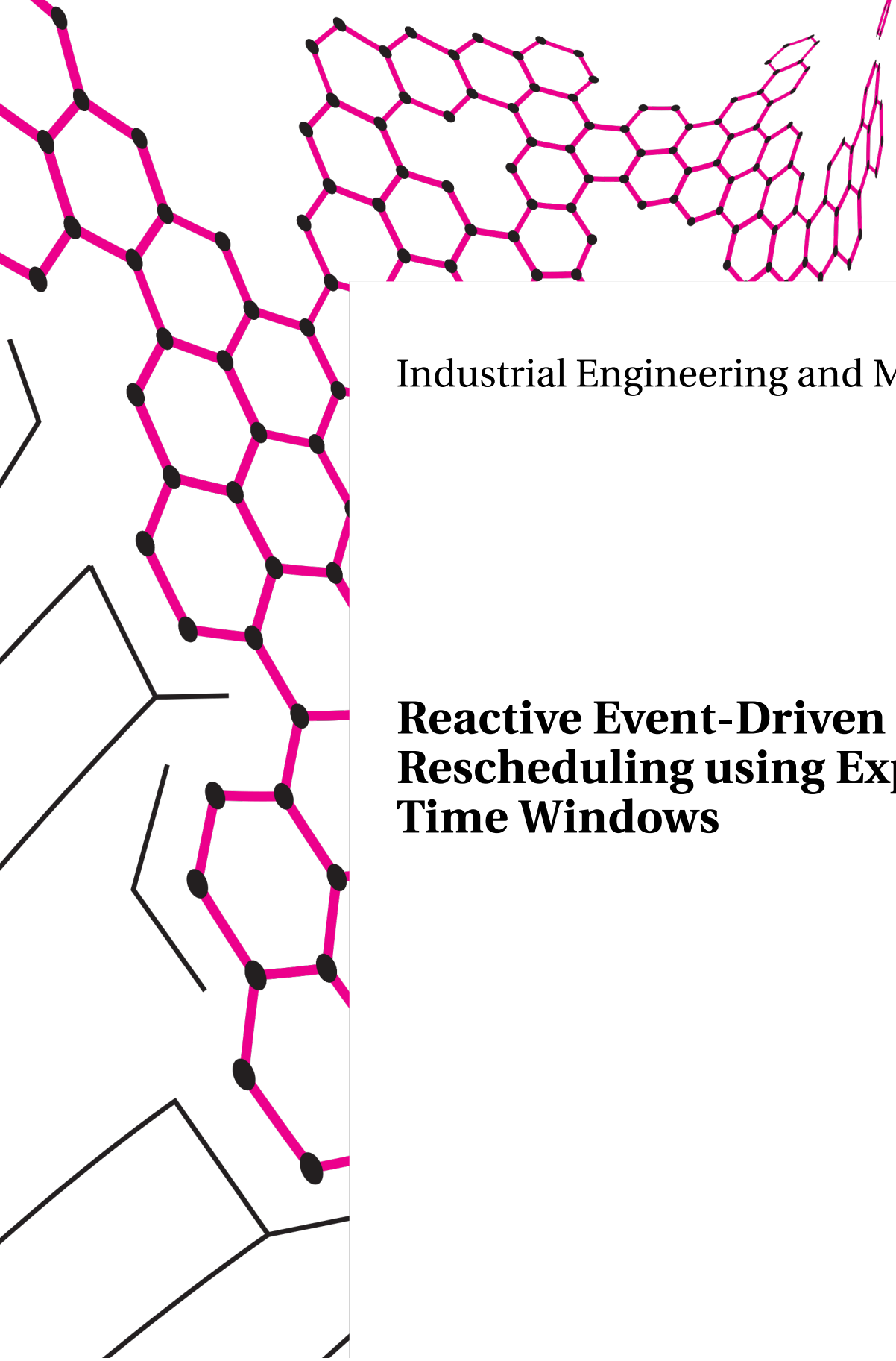


Industrial Engineering and Management

Reactive Event-Driven Rescheduling using Expanding Time Windows

htm aerotec

UNIVERSITY OF TWENTE.

Kim Buursema
June 20, 2025

University of Twente
Industrial Engineering and Management

PO Box 217
7500 AE, Enschede
Tel. +31(0)534899111

Reactive Event-Driven Rescheduling using Expanding Time Windows

Master thesis

Kim Buursema

First supervisor: Dr. Engin Topan
Second supervisor: Dr. Ir. Eduardo A. Lalla
Company supervisor: Arthur Kasteel

June 20, 2025
This report was written as part of the thesis assignment of the master Industrial Engineering and Management educational program.

Preface

Dear reader,

This thesis represents the final milestone of my Master's degree in Industrial Engineering and Management. It has been one of the most challenging and enriching experiences of my academic and professional development so far. The project, centred around predictive-reactive event-driven rescheduling using expanding time windows, allowed me to dive deep into a highly dynamic and practically relevant area of scheduling and optimisation.

The work was carried out in collaboration with HTM Aerotec, and I'm especially thankful for the chance to contribute to a real-world application where the outcomes of this research could have a tangible impact. It was both motivating and rewarding to apply theoretical knowledge in a practical setting and to see how abstract ideas can translate into tools and methods that support planning and scheduling in real operations.

I would like to sincerely thank my company supervisor, Arthur Kasteel, for his continuous support, feedback, and encouragement throughout this process. I also extend my thanks to my colleagues at HTM Aerotec. Their insights helped me keep the project relevant for use at HTM Aerotec, elevating the practical contribution of this research.

I'm equally grateful to my first university supervisor, Engin Topan. After reaching out to him, he suggested this research to me and connected me with Arthur. His guidance, constructive criticism, and thoughtful suggestions were essential from the early stages of the thesis, from shaping the research direction to refining the final results. I would also like to thank Eduardo Lalla for his enthusiasm and insights in the later stages of the research.

Finally, I want to thank my family, friends, and everyone who supported me throughout my entire studies. Their encouragement and support made a big difference during my studies, while their company and the fun moments together made my entire study period a great time to look back at.

I hope you enjoy reading this thesis!

Kim Buursema
Enschede, June 2025

Management Summary

HTM Aerotec manufactures high-precision components for the aerospace and defence industries. Each part must meet strict industry standards and undergoes multiple production steps involving both manual labour, performed by operators, and automated labour, executed by machines. Efficient scheduling is essential for timely delivery, but disruptions such as machine failures, operator illness and emergency orders frequently cause infeasibilities. Currently, HTM Aerotec lacks a practical and resilient scheduling solution to address such disruptions in a reasonable time frame.

To mitigate infeasibilities in the production schedule of HTM Aerotec after disruptions, we propose an event-based rescheduling model using time window expansions. With the proposed model and implementation, the required manual planning tasks are minimised, and disruptions can easily be mitigated.

Problem identification
In this research, we address the lack of a practical and resilient rescheduling process after disruptions. Practicality is defined by ease of use for the operator and planners, and short computational runtimes. The core challenge is the inability to quickly generate feasible schedules after disruptions due to excessive computation times in the current scheduling system by Van Boxel (2024). Other contributing issues are limited real-time data, operator availability, and tool constraints. Scheduling in this research context is complicated by constraints like fixture availability, operator shifts and a varying amount of manual and automatic labour per product type. To effectively address this, we formulated the main research question as follows:

> *How can a scheduling algorithm be effectively designed to mitigate scheduling infeasibilities after disruptions and minimise tardiness in production schedules for 5-axis machines within a reasonable time?*

We classify our problem as a single-machine rescheduling problem within a Flexible Machining Cell (FMC). Based on our review, only limited research is available on our research problem. Due to the unpredictability of our problem, we adopt a reactive scheduling strategy, making event-driven adjustments in response to disruptions. To manage computational runtime effectively, we incorporate an approach designed by Kuster et al. (2010), namely expanding time windows, to balance short runtimes and performance. This makes the approach practical by ensuring short computational runtimes.

Solution design
To solve our research problem, we developed a Mixed Integer Program (MIP). The MIP is an adapted version of the MIP of Van Boxel (2024), as our MIP can be used for rescheduling. The difference in the MIP thus lies in the set of operations to schedule and the input. Our MIP uses an initial schedule, and the set of operations to reschedule is based on a time window. Additionally, our MIP minimises schedule instability and average tardiness. We define a schedule as stable when the sequence of production is unchanged. Thus, the number of operations moved in the sequence of production is the schedule instability. However, an MIP is still too computationally intensive. Therefore, we designed a best-fit-based insertion rescheduling strategy. With an insertion strategy, operations to reschedule are inserted in a position in the production sequence. To determine what the best-fit position is to insert an operation, we first used a tardiness-stability estimation. This method estimates the stability by calculating the number of operations moved when inserting the operation at that position plus the estimated tardiness for that position based on predicted ending times for production. This is an estimation due to computational runtime limits. As a result, the best fit is not always optimal. Therefore, we developed three other methods, which can find more optimal solutions based on the research context. These are similar timeslot, ratio of hours left, and the weekly ratio balance method. The similar timeslot method scores potential insertion positions

based on their similarity, the timing during the day and week, with the old time slot. The ratio of hours left is based on the ratio of manual labour compared to the automatic labour hours in the day. This method scores positions based on how much this ratio corresponds with the manual to automatic labour ratio of the operation. The last method, the weekly ratio balance method, used the same ratio. However, this method scores based on the ability of that position to balance the weekly ratio.

The algorithm determines the operations to reschedule based on a time window. All operations which (partially) fall within this time window are rescheduled. The initial time window is determined by the duration of the disruption. At the end of each iteration, the time window size is expanded by increasing the upper bound of the time window. This is done based on one of four methods: linear, exponential, logarithmic or percentage-based. This creates an expanded set of operations to reschedule for the following iteration. The algorithm terminates if the maximal computational runtime or number of iterations has been reached. Using the expanding time windows, the model can effectively balance between finding close to an optimal solution while adhering to computational runtime constraints. This model is capable of effectively finding a new schedule which is feasible after disruptions within a reasonable runtime.

Experiments
We validated the model on two machines with realistic planning data, simulating 30 disruption scenarios. The instances we used have 3921 and 3874 operations to be planned, divided over 251 and 273 days, respectively. The scenarios generated with these instances have a disruption at a random moment in the planning horizon, at which one or multiple disruptions occur. In these experiments, we tested the best-fit method, the expanding time window method and the effect of the computational runtime limit. We designed four best-fit methods: tardiness and stability, similar timeslot, the ratio of hours left and the weekly ratio balance. First, we showed an insensitivity to the weight parameter ω for the tardiness and stability estimation due to the quality of the estimation, which is substantially affected by the presence of shift constraints. In the absence of the shift constraint, this method can be powerful to balance tardiness and stability. Among the four methods, the similar timeslot method achieves the best balance between tardiness, stability and utilisation for HTM Aerotec. However, in situations where stability is the main concern, the weekly ratio balance method should be considered. The time window expansion method experiments revealed that for HTM Aerotec, the percentage-based method is preferred due to the balance between tardiness and stability. Lastly, for the same reason, a runtime of sixty seconds is chosen. A sensitivity analysis of the due dates reveals that the only influence can be found in decreased tardiness, which is inherent with relaxing due dates. Moreover, the job mix only influences the utilisation of the machines.

The model's effectiveness is most present for disruptions which occur early in the planning horizon. At those moments, we see a utilisation rate of 75%. This is compared to a reference utilisation of around 90%, based on the initial schedule prior to the disruption. This schedule is no longer feasible and, therefore, is used as a theoretical upper bound to benchmark the performance of our model. While the model does not reach the efficiency of the initial (disruption-free) schedule, it does effectively provide the company with a feasible schedule which adheres to constraints and planning norms. Additionally, when comparing the model to a manually made schedule, we see that most operations generally considered as night and weekend tasks are planned during those moments, with only small improvements possible. Furthermore, we concluded that the model is generalisable to other flexible machining cells within similar production contexts. Also, if disruptions occur at the start of the planning horizon, the percentage-based method is preferred over the other methods. Lastly, if shift constraints are absent, the tardiness and stability-based scoring method is suitable.

Conclusions and recommendations
In this research, we present a practical and time-efficient rescheduling model for 5-axis machines at HTM Aerotec. The model is designed to reduce the impact of disruptions, such as machine breakdowns, by relieving manual work for the planner and providing quick solutions. The model uses percentage-based expanding time windows and a similar timeslot-based best-fit method to effectively balance schedule stability and tardiness within a computational runtime limit of sixty seconds. The stability of the method is

an average of 82 and 59 jobs moved for machines 538 and 539, respectively. Additionally, the tardiness reached is 5.5 and 0 hours with a utilisation of 61.1 and 41.1%. Disruptions in earlier moments in the time horizon lead to higher utilisation percentages, averaging at 75%. Thus, experiments show that once the initial schedule becomes infeasible due to disruptions, the rescheduling model performs effectively. The model was also tested on a second five-axis machine, showing similar performance and confirming its applicability to other FMCs within the organisation.

To support implementation, ERP system and fixture database connections were established to support model execution with real-time data. Additionally, we recommend that HTM Aerotec uses a new scheduling process: generate a new initial schedule bi-weekly, and if disruptions occur, follow a structured rescheduling procedure using the proposed model. Furthermore, we recommend involving key stakeholders during implementation, validating assumptions such as the labour time multiplier, and assessing the impact of changing this scheduling process. With these steps, we can ensure an effective implementation of the planning process.

Table of Contents

1| Introduction

1.1 The company

HTM Aerotec, a part of HTM Technologies, specialises in manufacturing high-precision components for the aerospace and defence industries. Known for its expertise in delivering technically complex and high-quality products, HTM Aerotec focuses on meeting the specific needs of each client, which can vary widely depending on project requirements. These parts are custom-engineered, often introducing varying levels of complexity and risk throughout the production process.

Each new part begins with the creation of a prototype, which undergoes rigorous testing and quality control to ensure it meets strict industry standards. Once validated, certifications are obtained, and the full batch can proceed to production. The company employs advanced manufacturing techniques, such as CNC machining and additive manufacturing, to ensure each part meets the required precision. Given the high variability in part requirements and the need for timely delivery, efficient scheduling is required.

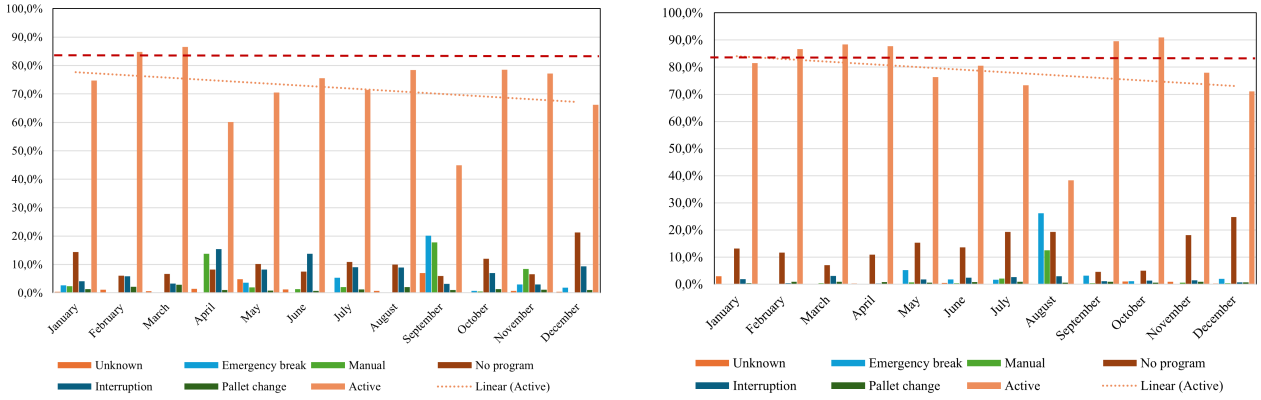
HTM Aerotec's commitment to quality is reinforced by continuous research and development, ensuring the company stays ahead of technological advancements in the aerospace and defence sectors. Through this approach, HTM Aerotec delivers reliable, high-performance components that adhere to the highest safety and regulatory standards, making it a trusted partner for clients in these demanding industries.

1.2 Problem identification

In the research of Van Boxel (2024), a start was made to bridge the gap between the achieved production hours of on average 105 and the theoretically possible production hours of 168. This resulted in a static scheduling algorithm improving the performance of the machines by at least 5%. However, this research also showed some topics for further research and in some cases, the scheduling algorithm was infeasible. Further research should validate the practical feasibility of the proposed solution. As the organisation responsible for improving machine utilisation and ensuring efficient production planning, HTM Aerotec is the problem owner in this research. These challenges directly affect the company's efficiency, workforce sustainability and service reliability.

1.2.1 Action problem

An action problem is a gap between the norm and the reality (Heerkens & Van Winden, 2017). In this case, this regards the production hours of the 5-axis machines at HTM Aerotec. The norm is to produce as many hours as possible, with the goal of 140 hours (83%)in a week, while maintaining a feasible workload for the company's planners. The goal of 140 hours is 28 hours less than theoretically feasible with the machines. However, achieving a 100% efficient production schedule is practically infeasible, due to disruptions like machine breakdowns. Furthermore, the production planning of the 5-axis machines is complicated by several aspects, including shift times of operators, unique fixtures required for production and the maximum number of tools available. This high complexity makes manual planning for the 5-axis machines laborious, which is a general problem for all similar machines (Slomp & Gupta, 2024). Section 2.5 further highlights these aspects and how these complicate the planning. In the past years, the machines have operated on average around 105 hours per week (Van Boxel, 2024). This is a significant gap compared to the possibilities, considering that 24/7 production theoretically is feasible, as unmanned production is possible. In the past months, improvements have been made to these production numbers. However, these improvements still fall below the set target of 83% as can be seen in Figure 1.1. These improvements have been possible due to elaborate communication between the production manager and planner. The planner has been required to invest greater effort in creating the schedule and rely more heavily on their



(a) Machine 538

(b) Machine 539

Figure 1.1: Comparison of Machine Status 538 and 539 in 2024

expertise regarding the products and systems. Consequently, their workload has increased. Without addressing this issue, the company risks declining production efficiency, increased risk for planner burnout, and potential disruptions in meeting client demands. Therefore, the action problem is formally defined as:

> *HTM Aerotec lacks a practical and resilient scheduling solution to address disruptions of an initial schedule and bridge the gap between the norm of 168 production hours per week on 5-axis machines and the current reality of 105 hours.*

1.2.2 Core problem

There are multiple causes for the inability to produce full-time without overburdening the planners. The problem cluster, Figure 1.2, visualises the connected issues, with three core problems in orange and the chosen core problem in red: machine malfunctioning during unattended periods, lack of real-time machine data, tool unavailability, and lengthy runtime of the static scheduling algorithm. A core problem is a problem that does not have any (known) cause and therefore is at the core of the action problem (Heerkens & Van Winden, 2017). Below, these core problems are explained.

The first core problem is the malfunctioning of the machine when operators are not around. Defects and errors within machines are inevitable; even with a lot of maintenance, these will occur at a certain rate. As an example, Figure 1.3 shows a moment where the machine broke down during the night, causing downtime for at least nine hours. To limit downtime, the operator needs to take immediate action to repair the machine. However, if the operator is not present, e.g. because it is on the weekend or during a break, then it will take a certain amount of time before the operator knows and solves the issues. This causes longer downtime for the machines. Due to the stochasticity of these machine breakdowns, we do not know when these breakdowns occur. Therefore, predicting when machines will break down and planning to have an operator present at those moments is not possible. Thus, this is not the focus of this research.

The second core problem is the fact that there is no lifetime data on machine status available. This problem is closely related to the malfunctioning, as the machine status shows to an operator if the machine is malfunctioning or not, or if it is empty. Because real-time machine data is unavailable to the operator when off-site, immediate responses are not possible. This again leads to longer downtime of the machines. A solution is researching methods to connect the machines to a system to show the lifetime data of the machine status. This data can then be used by operators, for example, during weekends. If the data shows that the machine is down, an operator can go to the company and solve the issue. That would improve the production times, as machine breakdowns are currently only noted once an operator is present. This system would require a reactive solution approach, as the data will show breakdowns at the moment they

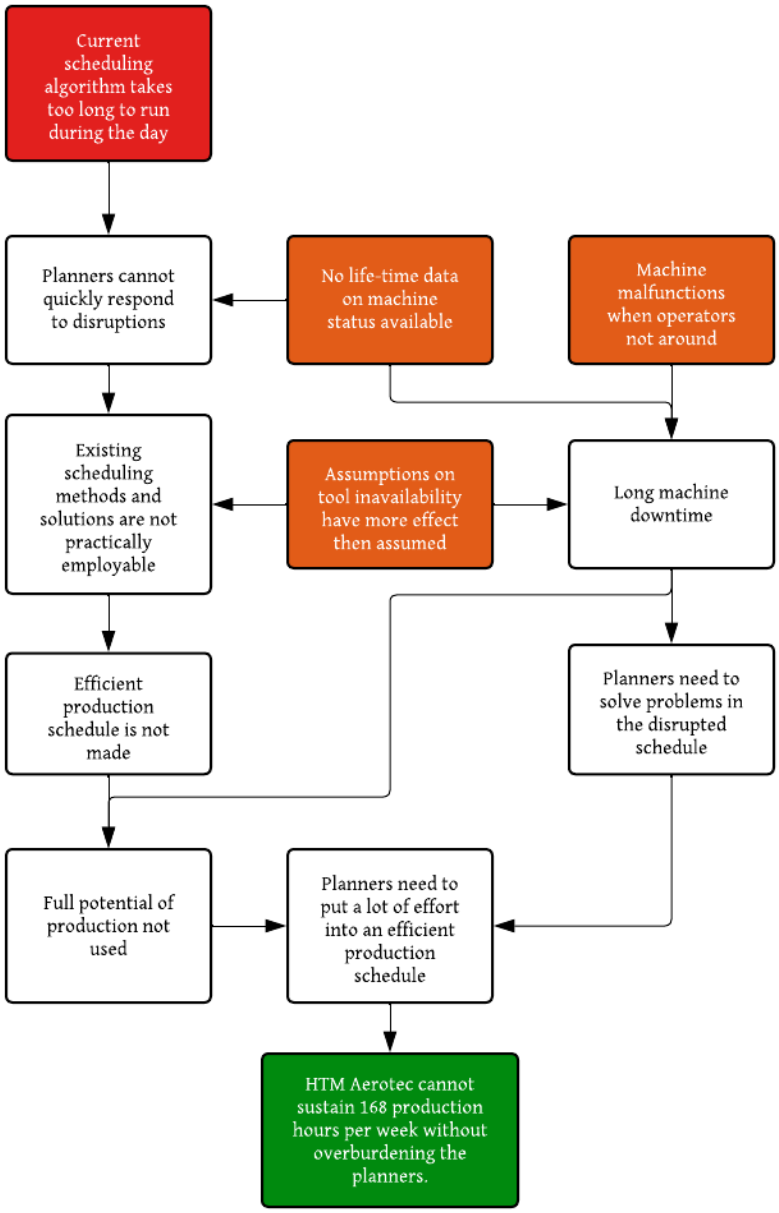


Figure 1.2: Problem cluster of HTM Aerotec, showing the causes of the action problem

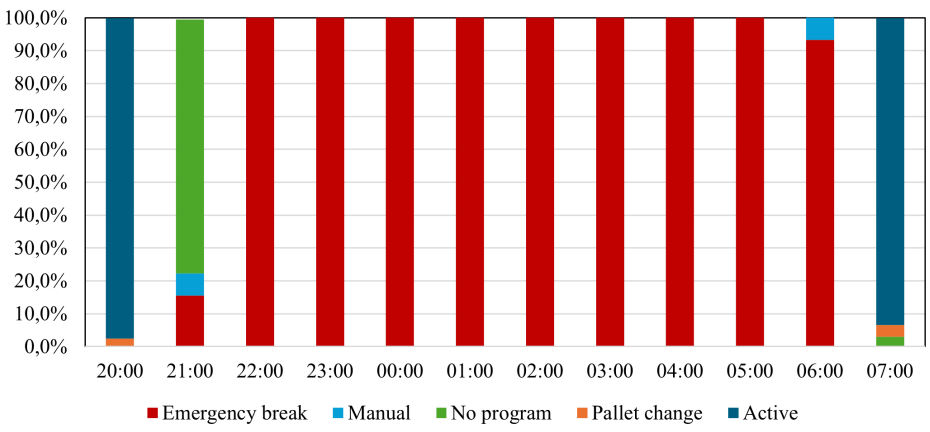


Figure 1.3: Example of failure machine 539 during the night

occur.

Next, in the past period, it was found that multiple times the machines were idle due to tool unavailability. This issue arises due to delays in tool maintenance or delivery and a capacity limit in the tool holder of the 5-axis machines (Section 2.2). Therefore, this influences the resulting schedule, leading to unexpected delays in production. A connection to the tooling database is possible. However, establishing this connection and obtaining the correct data is outside of the scope of this research. Therefore, this is not the chosen core problem.

Lastly, Van Boxel (2024) developed a scheduling model to provide the company with a schedule. This schedule can be used until disruptions occur (e.g. machine failure, change in due dates). This model is a static single-machine scheduling model; the input used is not stochastic, and the model produces a schedule for all items to be produced on one machine. However, the model is computationally demanding. Consequently, a new schedule resulting from the model would take too long to compute. As a result, operators and planners must manually intervene when disruptions occur to adjust and repair the schedule. This problem is shown at the top of the problem cluster. The rescheduling process after disruptions is discussed in Section 2.1. This process leads to a loss of production hours. Moreover, the workload of the planners becomes higher in these cases, putting more pressure on the planners. Thus, we can conclude that there is a need for a reactive approach, improving the current reaction process after disruptions. This research addresses that problem by exploring an approach that HTM Aerotec can implement in practice.

1.3 Research Design

In this research, we want to find a solution to the selected core problem. To find a solution, we formulate the research question as follows:

> *How can a scheduling algorithm be effectively designed to mitigate scheduling infeasibilities after disruptions and minimise tardiness in production schedules for 5-axis machines within a reasonable time?*

In this research question, multiple parts need to be researched to find an appropriate solution. Therefore, we set up sub-questions to structure the research process.

1. *What does the current scheduling situation look like for the 5-axis machines?*
   This sub-question has two components.
   (a) *How is scheduling currently performed for 5-axis machines, and what are the associated constraints and challenges?*
   (b) *How does the proposed solution by Van Boxel (Van Boxel, 2024) address these challenges, and what opportunities does it present for improvement?*

   The current scheduling situation can give insight into how a solution should be designed. We have already concluded that a (partially) reactive approach is needed. Further information is needed to ensure that all constraints and opportunities for a new solution are known. Therefore, this also serves as the base knowledge needed for sub-questions two and three. The question will be answered using a case analysis, aiming to find the current process and its constraints. Furthermore, the proposed solution of Van Boxel can give insight into the opportunities for the process of constructing schedules, mainly regarding a schedule before disruptions. These questions are answered in Chapter 2.

2. *What classes of scheduling approaches, supported by literature, are effective in accommodating and recovering from production disruptions?*
   This question will be answered using a literature review, in Chapter 3. We will look at the literature to find different classes of scheduling problems and to classify the type of problem in this research.

Then, this knowledge is used to select appropriate models that can react based on disruptions. By using this selection, we ensure that the final model is based on proven scheduling approaches. Hence, this question is used as a basis for question three.

3. *What should a scheduling solution look like to repair a disrupted schedule?*
   To make a suitable model, we require a theoretical model including all constraints and assumptions. Consequently, with this question, we aim to find the mathematical model corresponding to our research problem.

4. *How should a scheduling model be designed that can adapt to disruptions within a reasonable computational runtime?*
   This question aims to identify the parameters and assumptions necessary to construct a feasible scheduling model. Therefore, this question is used to find the required input, the practical assumptions needed, and how to provide a feasible schedule within a reasonable runtime. The aim is to ensure that the scheduling solution is appropriately modelled and aligned with the real-world process.

5. *What experimental design and validation metrics are required to optimise and evaluate the solution's effectiveness for real-world application?*
   When we know how a model can be developed, we need to find the appropriate settings for this context. Therefore, experiments are needed. This question aims to find the appropriate experiments and execute them such that the final model can be used for the machines at the company. Furthermore, validation metrics are used to ensure that the quality of the solution is at a sufficient level.

6. *What technical, organisational, and procedural requirements are necessary to successfully implement the scheduling algorithm in a production environment?*
   The goal of this research is to find a solution that can be used by the company. To ensure that the solution can be used, we need to research what is needed for successful implementation. This entails connections to input data, the front-end used for running the algorithm, and the manner in which the output is presented.

7. *What are the key findings, practical recommendations, and future research opportunities derived from this study?*
   Lastly, conclusions are drawn based on the previous sub-questions. These, together with some recommendations, are given to HTM Aerotec to finally answer the main research question.

1.3.1 Research Scope

At HTM Aerotec, there are many different machines. However, in the main part of this research, only two of the 5-axis machines are taken into account, numbers 538 and 539. The reason for the choice of these two machines is that sufficient data is available. Other 5-axis machines do not have performance data. Furthermore, other machines, in addition to 5-axis machines, are currently not seen as a bottleneck, and therefore, the total throughput of products in the company would not increase when looking into the production planning of these machines. In the experimentation phase, Chapter 5, we will include the other 5-axis machines of the company to test the generalisation of the proposed model.

Next to that, only the scheduling of these machines is within the scope of this research. Other processes within the company, like the entire flow of orders coming in, are not taken into account.

2| Current Situation

This chapter will answer the first sub-question as defined in Section 1.3. The context of the research is given. The first sub-question is:

What does the current scheduling situation look like for the 5-axis machines?

In Section 2.1, we show the current scheduling process for the machines. Section 2.2 elaborates on the machine types discussed in this research. Then, Section 2.4 elaborates on the proposed solution of Van Boxel, which is generating a static schedule using the earliest due date (EDD) dispatching rule for the initial solution and a tabu search to improve this solution.

2.1 Current scheduling process

At HTM Aerotec, each order goes through a similar process with multiple phases. This starts with the order coming in; an order request is made by the client and reviewed by the company. An advantage here is that around 90% of the orders are known in advance. Thus, order forecasts can be used in production planning. These forecasts are on the number of items and the due date. These forecasts have a high reliability, as the differences between these forecasts and actual orders are small. Therefore, up to two years of orders are known in advance. Once an order is in, the order goes to the order acceptance phase. Here, details like the due date and product requirements are specified. Using this information, the preparation work is done. This includes getting a FAI certificate, showing the exact machines and processes one product goes through. Such a certificate expires if the product has not been produced on that exact process for over two years. Next, the production planning phase starts. Here, the product is planned together with other orders. With this planning, the order moves to the production phase and finally onto shipment. The scheduling process that falls within our scope falls under the production planning phase. However, due to stochastic events like machine failure, production planning can be disrupted. Subsequently, the planning can become infeasible, leading to taking additional scheduling steps in the production phase. Therefore, these two parts of the process are elaborated upon.

Figure 2.1 shows the planning process. As mentioned, most orders can be forecasted. Therefore, the planning can be made far in advance with a time horizon of multiple months. Once a new forecast comes in, this order is planned in available time between other orders or added at the end of the schedule. When adding a new order to the schedule, always two days of buffer time are used over the entire throughput time of the order. This planning is currently made by the planner in an Excel sheet called the schedule list. The planner bases this on data from the ERP system, such as production times and order characteristics. In this list, all orders are sorted by their due date. The schedule list is sent to the production manager once every two weeks, who discusses the list with the operators. Important to note is that the operators are responsible for the final decision of what orders to produce. Therefore, the final order in which orders are produced might differ from the made schedule list. Deviations here are usually based on the production times of products. For example, some products have very short production times and therefore are not suitable for production during unmanned hours. This is because of the maximum number of products that can be put in a machine at once. Short production times during unmanned hours would ensure that the machine is finished before a new shift starts. Therefore, the choice is made to produce orders with longer production times so that the machine can produce throughout the entire night.

This planning process is disturbed by disruptions. Disruptions fall into one of five categories:

1. Machine failure
   Figure 1.1 shows that both machines have used the emergency brake, as well as moments where no program was used. Such a machine status can be caused by machine failure. For example, the program used to operate the machine during production has a mistake with the starting point, causing

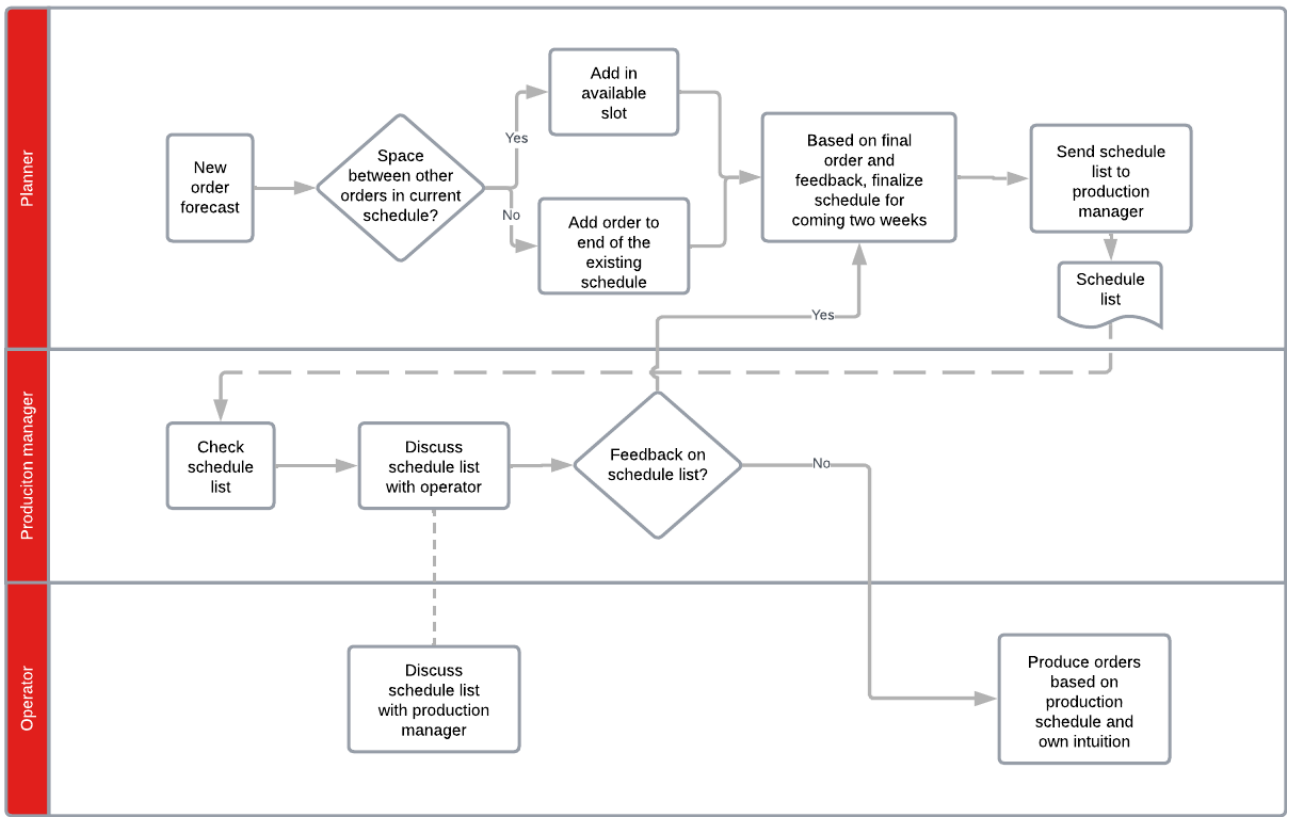


Figure 2.1: The process of adding new orders to the schedule

the machine to start milling into itself. This type of disruption typically leads to the job being processed to be unusable due to incorrect production. Therefore, we can define the start of this disruption as the start of the processing of the first operation of this job. The end of this disruption is when the machine is repaired and available for production again.

2. Operator illness
   If an operator is ill, this can cause extra downtime for the machines, as no new jobs are put into the pallet storage. However, due to the interchangeability of operators for different machines, the illness of one operator usually does not lead to extra downtime. This only occurs when multiple operators are absent. The start and end of this disruption are hence delimited by the time frame during which enough operators are ill to cause machine downtime.

3. Execution delay
   Execution delay of a job or order can occur for multiple reasons. First of all, jobs can require production steps on other machines before being ready for processing at one of the five-axis machines. If those other machines are behind schedule, jobs on the five-axis machines need to be processed later than planned. Another reason could be the need to wait for confirmation from the client or from the quality department. The start of this disruption is marked by the originally planned start of labour for this job. The end is the estimated moment at which it can be produced again.

4. Material unavailability
   Similar to execution delay, material unavailability leads to having to delay the start of processing jobs on the five-axis machines. This disruption starts and ends in a similar way as the execution delay.

5. Emergency order
   Emergency orders are orders which have a high priority and a tight deadline within a short time horizon. These do not necessarily disrupt a machine or cause downtime, but do disrupt the schedule and planning process, as these need to be planned within a few days. The start of this disruption is

the moment the emergency order comes in. However, as opposed to other disruption types, this disruption has no duration.

The process in case of disruptions is shown in Figure 2.2. Once a disruption is noticed, the production manager is notified. Together with the planner, decisions are made on changing the production schedule. Here, some different options exist. First of all, if the agreed-upon due dates with the customers can still be met, then the production schedule is not changed. This happens either due to a buffer or because the choice was made to start producing early. The production will continue when the machines are available again, based on the initially generated schedule. Often, that is not possible. For some orders, there is the possibility to produce part of the order and deliver the last part at a later point in time. In that case, the order is split up to free up capacity and ensure that due dates for all orders are met. When choosing this option, it is important to consider that orders must meet the minimum order quantity. Below this quantity, there are extra production costs. Another option is to move some orders to another machine. However, this is a lot of effort and depends on the available FAI certificate. Once a decision is made on the solution, the planner needs to update the schedule. In the current situation, the planner does this manually, causing a higher workload. This part of the process is shown in the grey block.

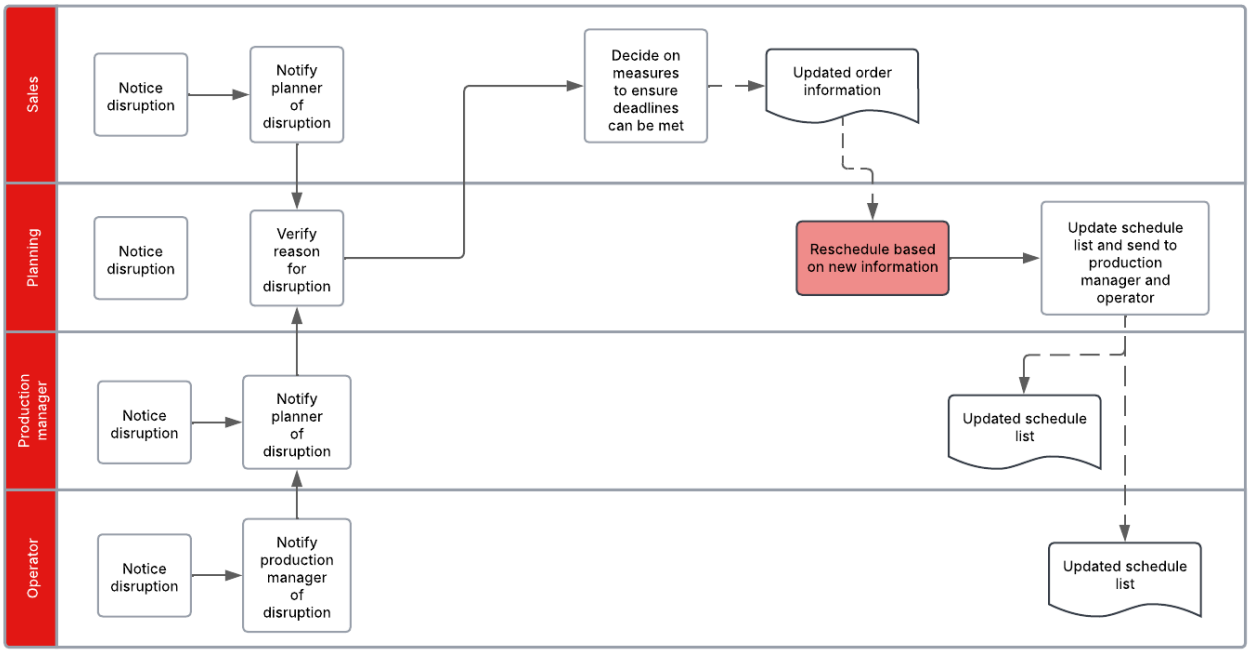


Figure 2.2: The process to deal with disruptions in the schedule (excluding new order arrivals)

2.2 The machines

As mentioned in Section 1.3.1, only two of the machines are within the scope of this research. These are 5-axis milling machines, shown in Figure 2.3. These machines are highly advanced and can be used for complex and precise machining tasks. The 5-axis regards the directions in which the tools can move. These are the three linear axes and two rotational axes. Due to these movement possibilities, these machines are efficient in producing materials with complex shapes and requirements. The products that HTM Aerotec produces are mostly parts that require high-precision and complex machining. Consequently, these parts require processing by at least one of the 5-axis machines.

The machines have an automated pallet handling system. This means that the machine has a storage compartment with pallets. On these pallets, the products are placed by an operator. The machine itself can take a pallet out of the storage compartment and start producing the product in the milling machine. This means that the machine can operate without an operator being at the machine. An operator is needed both to load new products into the storage compartment and to take out finished products, and to ensure

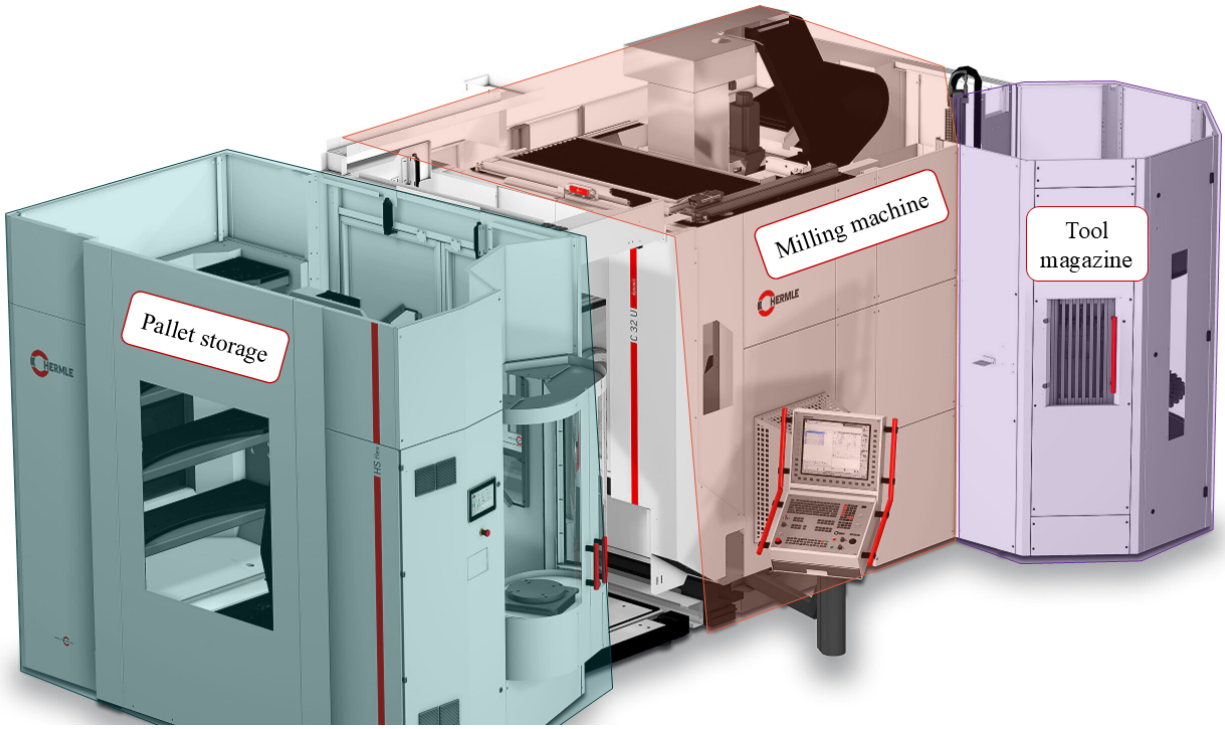


Figure 2.3: 5-axis milling machine and its pallet storage and tool magazine

Machine	# pallets	# tools
538	40	228 (192)
539	24	228 (192)

Table 2.1: Machine capacities

that the correct tooling is available in the tool magazine. Both machines have a set capacity for the number of pallets and tools, which is shown in Table 2.1. In the tool magazine, 3 pockets are always in use for three specific tools and 33 other pockets are reserved for the most used tools. Therefore, 192 pockets are remaining in both machines for other tools.

Products are loaded into the machine on pallets. These pallets are small surfaces on which products can be placed. An operator needs to place products on these pallets. The products themselves cannot be attached to the pallets, therefore requiring fixtures. A product is clamped on a fixture. The fixture is then assembled onto the pallet. The way this is done differs per product, and thus, the type of fixture needed can also differ. Some products, therefore, even require dedicated fixtures. Only a limited number of pallets and fixtures are available. In the case of dedicated fixtures, the number available is even lower than the general number due to the expenses needed to produce such a fixture.

2.3 Order processing and production considerations

Each order requires multiple parts of information throughout the process, such as the item-specific program needed for the machine, production requirements, and estimated runtimes. When an order is received, an initial production time calculation is made. A key part of the production times is the manual labour required, consisting of setup and labour. The setup time is the preparation required before the production of an item, including processing the first product. Labour time includes all tasks that need to be performed by the operator of the machine, including loading and unloading of items and other activities. These other activities, even though they do not occur at the machine, are included as they are essential work to keep the production process going. When scheduling production on machines, the labour hours that do not occur at the machine should be excluded, as they do not reflect the actual required time. This

is an adaptation that should be taken into account in the model.

After the initial estimation, the program used in the machine is created, which determines the required fixtures and tools. Currently, this information is stored across multiple databases, which are Glovia, Teamcenter and APM. The company uses an ERP system called Glovia. There, work orders, sales orders and a lot of other data are stored. Additionally, Teamcenter is a database for drawings, documentation, and production information. Fixture and tool information can also be stored there, but the storage method differs for different items. Lastly, APM is a database containing all information on tools (e.g. inventory, which production steps use what tool, etc.). Once an item is produced, this information is provided to operators. As the information is not stored in one database, this information cannot currently be used in the planning process and thus requires adaptation. In Section 6.1 we discuss the required changes for this research.

2.4 Static scheduling using Tabu Search

This entire section describes the research of Van Boxel (2024), who designed a scheduling tool using tabu search. Van Boxel's research aimed to optimise scheduling and increase production hours, which averaged 105 hours per week. This research focused on the same two machines, 538 and 539, as described in Section 2.2. Van Boxel created the scheduling method to optimise machine utilisation, thereby increasing production hours.

Van Boxel defined the initial scheduling problem as a single-machine scheduling problem with extensions. He chose the single-machine classification due to the FAI restrictions mentioned in Section 2.1. After obtaining an FAI certificate for a product, the product can only be produced on the certified machine. The initial time investment of obtaining a FAI certificate can be significant. Additionally, rerouting an item to other machines is a significant disruption in the entire internal supply chain. Therefore, we adopt the single-machine definition. Extensions in the research of Van Boxel are based on the pallet-handling system and follow the methodology used in the paper by Shin et al. (2020). These extensions enabled Van Boxel to define the problem as a Mixed Integer Linear Programming (MILP) problem. However, the large instances used at HTM Aerotec caused the runtime of this problem to become excessive. To address this, Van Boxel categorised the scheduling problem as NP-hard and opted to use heuristics instead of a MILP. The objective in the research of Van Boxel is twofold. First, he aims to increase the production hours per week for the machines. The production hours are directly reflected in the makespan, so he minimises the makespan. Secondly, he aims to minimise the tardiness.

Van Boxel began by constructing an initial schedule. To achieve this, he researched three dispatching rules: EDD, multi-factor, and random. The EDD dispatching rule multiplies the job's due date by a random factor to calculate a score, scheduling jobs in decreasing order of scores. Multi-factor combines multiple dispatching rules. In Van Boxel's research, the due date, number of fixtures available, and total hours contributed to a final score, which again determined the order of scheduling. Lastly, the random dispatching rule schedules jobs in a random sequence. In his experiments, Van Boxel found that the EDD dispatching rule produced the best initial schedule. The EDD rule resulted on average in an improvement of the objective value of 40-50% compared to the other two dispatching rules. In the larger data instances he used, this was obtained in seven minutes, and the smaller instances in two. He used input from an Excel file provided by the company's planners, which contained details on the jobs, their due dates, and related data.

Van Boxel then used meta-heuristics to improve the initial schedule. He experimented with tabu search (TS) and simulated annealing (SA), as well as two neighbourhood structures: variable neighbourhood (VN) and random. Through these experiments, Van Boxel determined that TS combined with a VN structure yielded the best final schedule. The TS relied on a tabu list to store neighbour solutions. Van Boxel set the maximum number of iterations to 125 and the tabu list length to 10, which significantly influenced the runtime and the final solution. Using the VN structure, Van Boxel designed operators to progressively

adapt the schedule, starting with a swap, followed by a move, multiple moves and finally moving all jobs.

Using EDD for the initial schedule and TS for improvement, Van Boxel demonstrated at least a 5% increase in production hours for both machines compared to real-life performance data. His sensitivity analysis identified several factors affecting the model's performance. These factors included operator visits during the weekend, shift lengths during the week, and the number of pallets and fixtures. Based on this analysis, Van Boxel showed that longer operator visits during the weekend and extended weekday shifts significantly improved performance. He also identified that shorter shifts were undesirable. In the current setup, operators' weekend shift lengths depend on their workload, making shift length a variable to minimise rather than a constraint.

Van Boxel implemented his solution using an algorithm written in Python. However, at the time of his research, most employees at HTM Aerotec lacked Python expertise. To address this, Van Boxel created an Excel interface connected to Python, enabling the company to execute the program. The program generated a graph indicating when to schedule jobs on both machines. Despite its utility, the program required a long runtime, rendering it unsuitable for handling disruptions. Consequently, the company does not currently employ this solution due to the infeasibility of its schedules and potential delays. However, the schedule is an effective initial schedule which could be used in dynamic rescheduling.

In his research, Van Boxel provided multiple recommendations for further study. Our research aligns with one of these recommendations: exploring dynamic rescheduling of production planning. This approach requires real-time machine data, which is unavailable. However, the Fraunhofer Innovation Platform has researched digitisation at HTM Aerotec. The result of this research is an interface which can be used by the operators of the machines. The schedule can be imported, and comments can be made on the schedule. Here, there is the possibility of adding a functionality for rescheduling.

2.5 Process constraints

The entire planning process is subject to multiple constraints, concerning both the products and the planning itself. In Section 1.2.1, we discussed that these constraints can significantly complicate scheduling. In this section, we discuss the constraints on the entire process and show examples of how these constraints impact the current scheduling process.

- Employee working hours
  The machines can operate unmanned, but the loading and unloading of the machines, called manual labour, need to be done by operators. Therefore, those activities can only be done during the working hours of the employees. The company operates with one shift per day. These shifts are between 7:00 and 16:00, with a one-hour break in between. On weekends, the shifts are limited to a maximum duration of three hours for all machines combined, depending on the amount of work that is needed. Per machine, this comes down to 45 minutes. Furthermore, the times depend on the operator. These employee working hours provide us with a manual-to-automatic labour ratio. This ratio is the hours of manual labour compared to the total hours of labour (manual plus automatic). If we look at the available resources in one week for our research problem, this ratio is around 24.7%. This ratio indicates how easily the full potential of automatic production can be used. For example, if the jobs on a machine have a ratio of 30%, that means that more manual labour is required than the total shift hours we have. Thus, not all automatic production hours can be utilised.

- Max number of places in machine inventory
  The machines both have a fixed number of pallets available in the inventory cell where products can be placed to be produced. Products are taken from this cell, produced, and then placed back in this cell. Therefore, the constraint for the number of places in the machine inventory is on finished products, products still to be produced, and the product currently being produced.

- Number of fixtures available
  Fixtures are used to clamp products on the pallets. Some products require specific fixtures. Further-

more, there is a limited number of general fixtures. Therefore, the number of fixtures available for a product is a constraint.

- Multi-fixturing
  Some products can be produced together. This is done by putting them on a tower, which is then put on a pallet. There is a limited number of towers available, and this is only possible for some products. This does not influence the production times, as each product retains the same production times. However, this does partially relax the pallet inventory constraint by allowing more items in the machine.

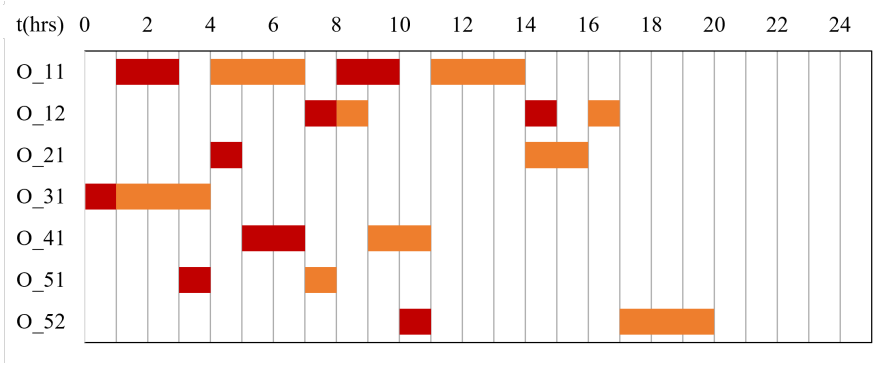
- Release and due dates
  Each work order has a given due date which needs to be met. Furthermore, release dates are added to simulate the preceding steps. If these steps are not finished, the release date is pushed back.

- Production steps within one machine per item
  Some products require multiple production steps within one machine. These steps have a specific order, but these steps are listed as separate jobs. Therefore, these have a precedence constraint.

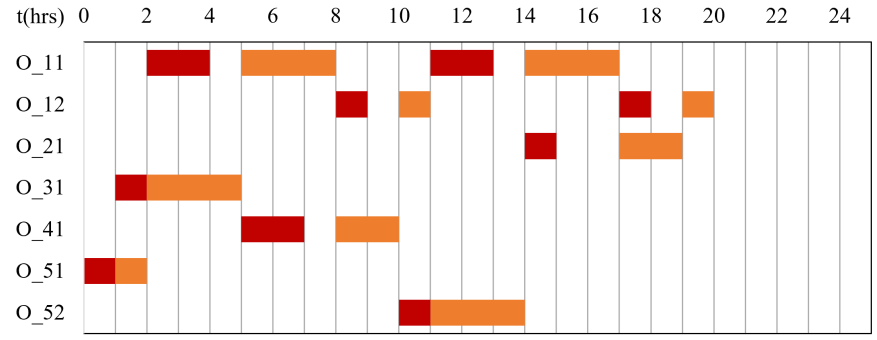
- Runtime limit
  As mentioned in Section 1.2.1, the company aims for high production hours per week. If rescheduling is needed, for example, due to machine failure, then a new schedule is needed within minutes. This gives a constraint on the runtime of the rescheduling algorithm. A strict constraint is not set. However, it should be within a maximum of a few minutes.

Figure 2.4 illustrates a basic example of a production schedule. We refer to operations using the notation O_{ji}, where j is the job identifier and i indicates the operation number within that job. For example, O_{11} is the first operation of job 1, and O_{52} is the second operation of job 5. Job identifiers (e.g., 1, 5) are arbitrary and do not imply any specific processing or scheduling order. In this scenario, we have one machine with two places in the pallet inventory. Furthermore, we have precedence constraints for the items requiring two production steps, jobs 1 and 5. Each production step first requires manual labour and machine processing. Figure 2.4(a) shows one of the optimal solutions in terms of makespan if there are no additional constraints. However, in this solution we can see at for example t=8 there are four pallets in use (for O_{11}, O_{12}, O_{21} and O_{41}). However, due to limited pallet capacity, in this example, only two operations can be in the system simultaneously if they have started labour but not yet completed processing. As a result, any delay between labour and processing occupies a pallet space. Therefore, the first solution is not feasible anymore when we add this pallet constraint. Building a schedule for this situation can be done in multiple ways; however, it is already a challenging task to find an optimal one. Figure 2.4(b) shows an optimal schedule considering only the pallet constraint.

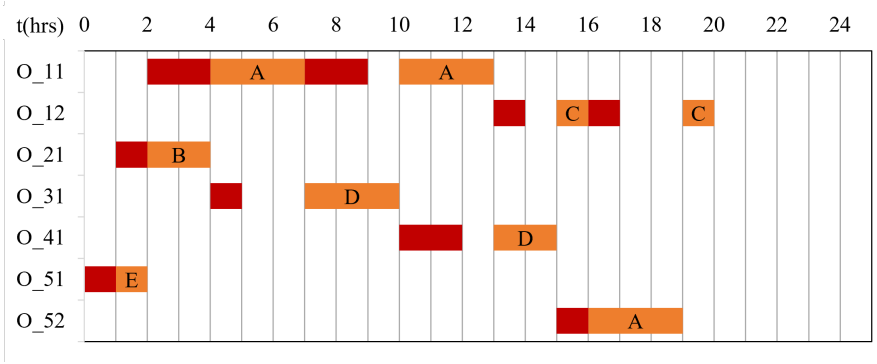
Next, we introduce dedicated fixtures. O_{11} and O_{52} require a fixture of type A, O_{21} type B, O_{12} type C, O_{31} and O_{41} type D and O_{51} type E. With these dedicated fixtures, the schedule shown in Figure 2.4(b) becomes infeasible; operations O_{11} and O_{52}, which share the same fixture, are in the machine simultaneously. Therefore, the schedule needs to be adjusted, as shown in Figure 2.4(c). Finally, we introduce a third constraint: labour shifts, which only allow manual work between periods 0-8 and 12-20. Again, the previous schedule becomes infeasible, for instance, the labour of O_{11} is planned outside of these hours. An updated feasible schedule is shown in Figure 2.4(d). The hours outside of the shift hours represent the night and weekend hours at HTM Aerotec, when no operator is present. This basic example demonstrates scheduling complexity when introducing multiple constraints. At HTM Aerotec, the scheduling problem involves many more items and operations, making scheduling significantly more difficult.



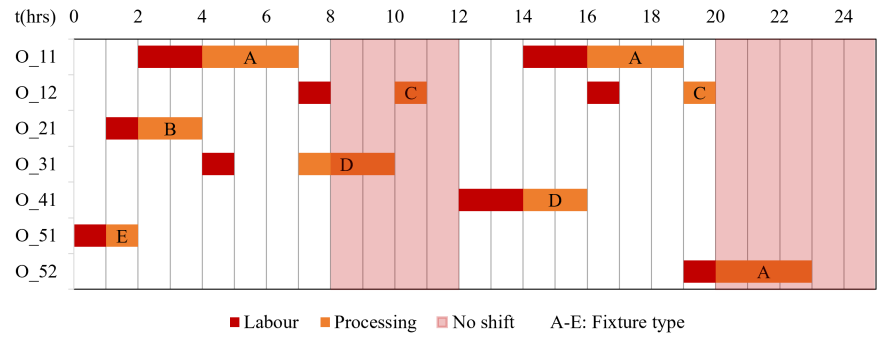
(a) No constraints



(b) Pallet capacity 2



(c) Pallet capacity 2 and dedicated fixtures



(d) Pallet capacity 2, dedicated fixtures and labour shifts

Figure 2.4: An example of the scheduling complexity with three constraints

3| Scheduling approaches

This chapter aims to answer the second research question by using a systematic literature review. The second research question was defined as follows:

> *What classes of scheduling approaches, supported by literature, are effective in accommodating and recovering from schedule disruptions?*

3.1 Type of scheduling problem

A job shop scheduling problem is a problem containing multiple products which require processing on different machines and in different orders (Zheng & Sui, 2019). Therefore, the routing of these items is of importance. However, in this research, we limit ourselves to the five-axis machines. The parts only need to be processed by one of these machines. Therefore, we can simplify this problem by omitting the route before and after these machines. The refined scheduling challenge can thus be classified as a parallel machine problem. As the machines are similar, where only some capacity constraints differ, we can further specify the machines as identical parallel machines (Kuo & Li, 2024).

However, as described in Section 2.4, Van Boxel characterises the scheduling problem as a single-machine Flexible Machining Cell (FMC) problem. This is due to the FAI requirements, causing a given machine per item. The same holds in our research, and thus we adopt the simplification to a single-machine scheduling problem. The FMC generally consists of at least one machining module, a loading robot and an inventory for both products and tools (Luo et al., 2022). Many studies on FMC planning and scheduling assume the presence of a conveyor belt supplying all products (Dawande et al., 2005; Hamasha et al., 2015; Zhou & Lei, 2021). Only in limited research on single machine problems, e.g. of Adil and Shaw (2024) and Shin et al. (2020), this assumption is not made. In this research, we do not have a conveyor belt with a steady flow of incoming products. Therefore, the scheduling problem in our research is an extension of the problems researched in Shin et al. (2020) and Van Boxel (2024). Multiple-machine FMC problems are discussed in the literature as well. However, in these problems, there is one cell containing multiple machines. Here, one robot arm transports products to one of the machines in the cell (Dawande et al., 2005; Sethi et al., 1992). We have not found literature discussing problems regarding multiple cells containing one machine.

3.2 Dynamic scheduling

Production schedules can be generated using different approaches. In theory, different algorithms are defined to make optimal or near-optimal schedules. Scheduling problems can be solved in three ways: static, dynamic and active scheduling (Y. Li et al., 2023). Static scheduling is the type of scheduling used by HTM Aerotec and Van Boxel (2024). This type of scheduling is unable to respond to uncertainties, causing disruptions. In practice, the final production schedule and its execution often deviate from the algorithmically generated schedules (Y. Li et al., 2023; Lou et al., 2012). Several factors contribute to these deviations. According to Vieira et al. (2003), the most common include operator decisions, machine failures, and other disruptions, such as those occurring at HTM Aerotec. This type of uncertainty can be defined as dynamic events. Dynamic scheduling or rescheduling (Holguin Jimenez et al., 2024; Wu et al., 2025) takes these disruptions into account and can repair a mistake or change in the schedule.

In the literature, there are many similar definitions of dynamic scheduling and rescheduling. Figure 3.1 shows the structure used in our research. In the paper of Vieira et al. (2003), dynamic scheduling is defined as a scheduling method which does not generate or use an existing schedule. However, they also mention other papers referring to dynamic scheduling with a definition using schedules. Sligting (2024) positions his research within the framework of Vieira et al. (2003) as a static, deterministic rescheduling environment using predictive-reactive strategies and focusing on schedule repair methods. Our research

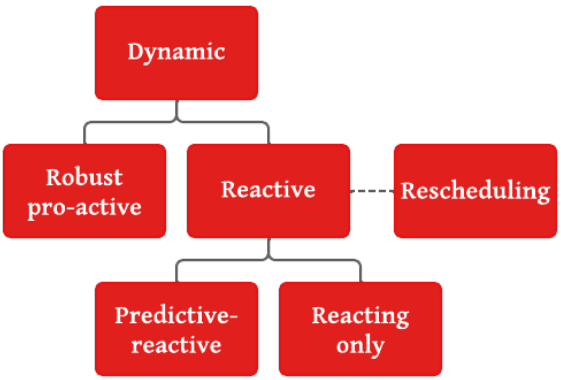


Figure 3.1: Scheduling strategies

has a similar environment and focus.

In contrast to the research of Vieira et al. (2003) and Sligting (2024), we define dynamic scheduling as the handling of dynamic events during the scheduling process. Van Boxel (2024) categorises dynamic scheduling into three categories: reactive, predictive-reactive and robust pro-active scheduling. He defines the reactive approach as responding to disruptions when they occur, without having an initial schedule. A predictive-reactive strategy elaborates on this, starting with an initial schedule. Then, after a disruption, this schedule is repaired (Tighazoui et al., 2021; Vieira et al., 2003). The robust pro-active strategy makes a schedule accounting for the disruptions beforehand (Shariatmadari & Nahavandi, 2020), consequently eliminating the need to react after disruptions. Schedule robustness is a method to build slack into the schedule, thereby minimising the impact of dynamic events (Wojakowski & Warzolek, 2014). Van Boxel (2024) showed how including robustness changes model behaviour. Including robustness changes the definition of his work to a robust, proactive method according to these definitions.

We deviate from these definitions due to the semantics in these definitions. In a scheduling context, the term ”predictive” suggests the prediction of dynamic events. Therefore, we define predictive scheduling as generating initial schedules based on predictions of dynamic events. Hence, we do not classify the inclusion of robustness as predictive. Reacting then occurs when the schedule becomes infeasible. Rescheduling is defined by Holguin Jimenez et al. (2024) as adjusting an initial schedule based on the occurrence of dynamic events. While rescheduling is commonly classified under dynamic scheduling, we can also argue that rescheduling should be defined as static scheduling. This is possible when the disruptive event has already occurred and full information is available. In such cases, the dynamic nature is resolved, and the scheduling tasks occur without uncertainty, making the problem static. However, the complete resolution of the dynamic nature is virtually impossible as uncertainty and changes are ongoing in real-world settings. Therefore, we adopt the classification commonly used in the literature. Thus, the process of rescheduling falls within the scope of dynamic scheduling and, more precisely, is an aspect of reactive scheduling.

Reactive strategies are particularly valuable when proactive or predictive scheduling is complicated by limited data availability or unknown parameters to define stochastic processes (Lou et al., 2012). The problem in this research requires responding to disruptions after their occurrence. Furthermore, we are unable to define the stochastic process of the dynamic events. However, as a result of the research of Van Boxel (2024), we do have an initial schedule to work with, which includes schedule robustness. This initial schedule has some limitations that render it infeasible to use. When combining this with rescheduling, these infeasibilities could be seen as disruptions. Accordingly, in this research, we adopt a reactive scheduling strategy.

3.3 Solution approaches

In this section, we review different solution approaches for rescheduling. We review event-driven and periodic rescheduling, analyse repair methods and define performance measures for rescheduling strategies.

3.3.1 Event-driven and periodic rescheduling

Within rescheduling, three different strategies are possible regarding when to start the rescheduling process. These are event-driven, periodic or hybrid (Vieira et al., 2000). Event-driven rescheduling is a strategy where certain events trigger the rescheduling process. Examples are job arrivals, surpassing a threshold for the queue length or machine failure. This strategy can effectively handle the arrival of sudden events, but can also lead to frequent rescheduling. Periodic rescheduling has set periods after which the rescheduling process is started. These periods are of a set length. Periodic rescheduling gives stability to the schedule. However, the arrival of sudden events is handled poorly. The hybrid strategy uses a combination of both, where each period triggers the rescheduling process as well as certain events.

Table 3.1 shows that eleven articles use an event-driven strategy. Within these articles, different types of events are mentioned as being able to trigger rescheduling. Examples are new order arrivals, due date changes or priority changes. Only one article in the literature uses periodic rescheduling. This is done by Pfund and Fowler (2017), who also compare this strategy to event-driven. They conclude that a periodic rescheduling strategy is most useful at the beginning of a scheduling horizon, whereas event-driven outperforms at moments later in the scheduling horizon or where the variability is larger. They refer to the variability of processing time, but we generalise this to the variability of the process, as machine breakdowns could be seen as causing high processing times. Nine out of twenty use a hybrid strategy, therefore partially using the periodic strategy. The reasoning for the use of this strategy is explained by these articles as taking the benefits of both periodic and event-driven strategies. It gives the stability of periodic rescheduling while giving the option for handling sudden events.

In this research, we do not require periodic rescheduling. We can use the model proposed by Van Boxel (2024) to periodically schedule all operations. Therefore, we opt for an event-driven strategy to effectively accommodate the arrival of sudden events. The possible frequent rescheduling and thus often changing schedule is an aspect which we need to minimise. We further explain measures to handle this in Section 3.3.3.

Table 3.1: Overview of articles and the discussed repair strategies

Article	**Scheduling Type**	**Applied Strategies**	**Key takeaway**
Elmaraghy et al. (1998)	Event-Driven	GA, dispatching rules	Due to requiring fast response times, dispatching rules are a good option for rescheduling compared to GA
Henning and Cerdá (2000)	Event-Driven	Interactive Scheduling	The time to make the schedule was reduced by almost eight hours, schedulers welcome maintaining control and input over the process
Kuster et al. (2010)	Event-Driven	Insertion-based	The local rescheduling is a good strategy for large, complex problems
J. Palombarini and Martínez (2012)	Event-Driven	DRL	This approach requires extensive simulation but is a good strategy providing stability and low tardiness

Continued on next page...

Article	Scheduling Type	Applied Strategies	Key takeaway
Hamzadayi and Yildiz (2016)	Event-Driven	Complete rescheduling (SA and DR based)	SA-based complete rescheduling outperforms DR based
G. Li et al. (2017)	Hybrid	Right Shift, Complete rescheduling, Insertion-based	Hybrid outperforms periodic if many urgent tasks arrive
Pfund and Fowler (2017)	Periodic, Event-Driven	LRH+	Periodic rescheduling is most suitable at the start of the planning horizon and/or with low variability, otherwise event-driven is more suitable
Baykasoğlu and Karaslan (2017)	Event-Driven	GRASP	Event-driven outperforms periodic rescheduling, GRASP is a suitable rescheduling strategy
Baykasoğlu et al. (2020)	Event-Driven	Order review release (ORR)	Using ORR, more orders can be accepted compared to using dispatching rules
Wang et al. (2020)	Hybrid	GA	The improved GA performs better and is faster than a simple GA
H. Zhang and Zhang (2020)	Hybrid	Multi-level based dynamic scheduling	Using rolling time windows, this is suitable for complex, highly changing environments
J. A. Palombarini and Martinez (2019)	Event-Driven	DRL	The approach uses knowledge of results of repair operations but is based on a Q-network, making it unsuitable for complex environments
Liang et al. (2019)	Hybrid	GA	A rolling time-window is effective in hybrid rescheduling strategies
B. Zhang et al. (2023)	Event-Driven	Complete rescheduling, GA	GA has better stability than baseline and right-shift strategies, and total costs decrease if stability is considered in rescheduling
Ghaleb and Taghipour (2023)	Event-Driven	Modified SA	The proposed SA outperforms dispatching rules and iterated greedy approaches. However, the runtime limit was set at 2 hours
Liu et al. (2023)	Hybrid	(partial) MILP	Good solutions in terms of stability, makespan, and tardiness. However, due to the use of MILP, it is unsuitable for large, complex instances
Cheng et al. (2022)	Hybrid	Insertion-based, GA	A rolling time-window strategy is useful in rescheduling, collaboration between human and machine rescheduling is effective
Yan et al. (2024)	Hybrid	(R/D)RL	The DRL model gives fast solutions, with high quality. However, implementation is complex

Continued on next page...

Article	Scheduling Type	Applied Strategies	Key takeaway
Usman Nisar et al. (2024)	Hybrid	Exact, GA, GRASP	Exact is only possible for very small instances, GA is suitable when fast solutions are required, GRASP outperforms both
This research	Event-Driven	Insertion-based, time window expansion	Due to a large, complex environment, requires strategies with short runtime

3.3.2 Repair methods

Various methods exist to repair a schedule during rescheduling. Table 3.1 provides an overview of the type of repair methods used by articles. The most straightforward method is the right shift. Here, all jobs are moved up the timeline to accommodate the rescheduling triggering event. This method is easy to implement but often does not result in efficient schedules. Nevertheless, this approach is commonly used as a baseline in studies developing more advanced repair methods (e.g. G. Li et al. (2017)). Insertion-based strategies share similarities with the right-shift approach. The articles describing insertion-based strategies mostly insert new order arrivals, or take the jobs that have changed details and insert those at feasible places. This can be combined with a right shift to make the insertion feasible. Orders can be inserted into the schedule using various heuristics; first-fit, best-fit and worst-fit are examples of such strategies. Best-fit heuristics aim to place operations in the smallest available time gaps they can fully occupy, minimising idle time and improving overall efficiency. This approach is different from a first-fit method, which assigns an operation to the first available position that meets its constraints. First-fit is computationally simpler, however, it can lead to suboptimal scheduling because earlier operations may be placed inefficiently. This leaves gaps that do not align well with future operations. Best-fit ensures that gaps are utilised optimally, reducing overall idle time and increasing schedule efficiency. In contrast, worst-fit adopts the opposite method of the best-fit method by always choosing the position that leaves the most idle time.

Kuster et al. (2010) extend the insertion-based strategies by using an expanding time window. An initial time window is calculated, within which all events are rescheduled. Then, they expand the time window and do this process again, until some stopping criterion (for example, a maximum runtime) becomes true. Time-window expansion can be achieved through three different approaches: linear, exponential, and logarithmic. In a linear expansion scheme, the time windows increase by a constant amount in each iteration, maintaining uniform step sizes (see equations 3.1, 3.2, obtained from Kuster et al. (2010)). l_0 and u_o are the initial lower and upper bounds of the time window. t_c is the current time, and t_h is the end of the planning horizon. M is the maximum number of time window expansions. As a result, we have Δl_m and Δu_m, which are the amounts with which the bounds change in iteration m. In contrast, an exponential expansion scheme results in step sizes that grow exponentially with each iteration. This approach prioritises moments closer to the original time window, as initial expansions are smaller, while later expansions become significantly larger. On the other hand, a logarithmic expansion scheme emphasises moments further from the original time window. This occurs because the initial step sizes are relatively large, but their rate of increase diminishes over time, leading to a progressively slower expansion rate. In addition to these strategies defined by Kuster et al. (2010), we propose a percentage-based expansion strategy. This strategy expands the size of the time window by a set percentage, therefore having exponentially increasing step sizes over time. This strategy is easier to comprehend compared to the exponential strategy. Moreover, in our context, we fix the lower bound at the current time due to the nature of disruptions. These adaptations to the methods of Kuster et al. (2010) highlight key distinctions to adapt our research to our context.

$$\Delta l_m = \frac{l_0 - t_c}{M} \tag{3.1}$$

$$\Delta u_m = \frac{t_h - u_0}{M} \tag{3.2}$$

Another method is complete rescheduling, which is mostly used in cases where either the problem instance is small or there is sufficient solving time. The latter occurs mostly in periodic rescheduling, where, at the end of a rescheduling period, sufficient time can be taken to run the model. A genetic algorithm (GA) evolves a population of solutions, selecting the fittest and applying mutation and crossover to find new solutions. This process is done until some stopping criterion becomes true. Wiendahl and Garlichs (1994) were one of the first to show the effectiveness of GAs in scheduling. This is later improved with, for example, local search (Wang et al., 2020), rolling windows (Liang et al., 2019), and collaboration between human and machine (Cheng et al., 2022). The Lagrangian Relaxation Heuristic (LHR+) (Pfund & Fowler, 2017) is used to combine scheduling with the dispatching of AGVs, which therefore is not of interest to our research. Some forms of reinforcement learning, like deep reinforcement learning (DRL) and relational reinforcement learning (RRL), are used as a smart way to optimise scheduling problems without leading to long computational times. However, reinforcement learning techniques are complex to implement. A modified simulated annealing (SA) algorithm is used by Ghaleb and Taghipour (2023). The normal SA is adapted to only partially reschedule. Lastly, Henning and Cerdá (2000) distinguish themselves within the rescheduling literature by combining human and computer knowledge. They show a dashboard to the operator giving feasible scheduling solutions and their results. With this dashboard, the operator can decide on the way to repair the schedule. They believe that this strategy ensures that operator knowledge and practical implications can be integrated into the rescheduling process.

In this research, we choose a local rescheduling strategy. We use expanding time windows to decide on the jobs to reschedule and use an insertion-based method to repair the schedule. We chose the insertion-based method due to its simplicity in implementation, combined with short computational runtimes. As opposed to Kuster et al. (2010), we do not change the lower bound of the time window. In this research, the lower bound is fixed at the current time due to the type of disruptions that occur. Therefore, we only expand the time window using the upper bound.

3.3.3 Rescheduling performance

The main objective of this research is to minimise tardiness. However, when introducing event-driven rescheduling, we determined that this can lead to many changes in the schedule. Moreover, when introducing rescheduling, some performance measures for the rescheduling strategy become essential. Therefore, in this section, we review rescheduling performance measures.

To begin, Van Boxel (2024) incorporates schedule robustness in his solution to give a higher likelihood of feasibility in practice. Van Boxel (2024) defines schedule robustness as the inclusion of a buffer as a percentage of the operation-specific production time. This causes the initial schedule to have a higher resilience. This schedule robustness presents in the form of a buffer in the planning, so the makespan is slightly higher. Van Boxel (2024) suggests further research regarding the impact of schedule disruptions on performance. In this research, the addition of schedule robustness in the initial schedule remains significant. In Section 3.3.1, we conclude that partial rescheduling is most likely to result in a worse schedule than complete rescheduling. However, complete rescheduling is only possible at limited moments. Furthermore, in practice, regularly needing to reschedule can have a large impact on the rest of the company and the supply chain within the company. Therefore, schedule robustness can help ensure that the need for partial rescheduling is limited and the schedule used in practice is close to optimal. Van Boxel has conducted a sensitivity analysis on the level of robustness to include. The results showed that only a five per cent robustness would already increase the objective value by almost eight per cent. Therefore, in implementation, the level of robustness should be analysed to find an optimal balance between objective value and the number of times rescheduling is needed.

Beyond the aspect of scheduling robustness, there are performance metrics relevant to rescheduling. These are different from general scheduling objectives like makespan. The performance measures have the goal of measuring the impact of rescheduling on the process. One important metric is scheduling stability. The scheduling instability, or nervousness, can be defined as the total deviation of the new schedule after rescheduling compared to the initial schedule (Qiu et al., 2024; Vieira et al., 2003). This deviation takes on many different forms, from due date changes and quantity changes to schedule changes (Pujawan, 2004). Stability and robustness are distinct concepts; while robustness aims to prevent disruptions by incorporating buffers, and enhancing the schedule's resilience, stability is a performance measure used after disruptions occur to assess how well the schedule adapts. Cowling and Johansson (2002) define a basic formula, shown in Equation 3.3, to calculate the schedule stability based on starting and completion times. The input of this formula is the model, or process, M, the initial and reactive scheduling strategies S_{static} and S_{react} and lastly the information E arriving at time t. Furthermore, t_i is the starting time, C_i is the completion time, D_i is the maximum possible disturbance, and α is a weight factor or cost of the disturbances. In this notation, the disturbance is defined as a cost variable, where it can be the cost of delay in terms of time or the cost of outsourcing. The decision for the value of D_i depends on the product, but this can be outsourcing, delaying or cancelling the order. Al-Hinai and Elmekkawy (2011) propose a similar measurement but uses weighted stability by dividing by the total number of operations. These measurements of stability take into account the deviation of processing times. However, Narayanan and Robinson (2010) identifies two other types of stability: the number of jobs rescheduled divided by the total number of jobs, and the time of the setup of one product family divided by the total number of product families. Similarly to the first, stability can also be measured by the deviation in processing sequence (Blackburn et al., 1985). In our research context, the processing sequence is important due to the nature of the supply chain surrounding the machines we are rescheduling. Therefore, we will define schedule stability as a schedule where the sequence of production is stable.

$$S(M, S_{\text{static}}, S_{\text{react}}, E, t) = \sum_{i=1}^{n} \min\{\alpha(|t_i - t_i'| + |C_i - C_i'|), D_i\} \tag{3.3}$$

3.4 Conclusion

In this chapter, we answer the second research question. We classify the problem as a single-machine scheduling problem in a Flexible Machining Cell (FMC) without the assumption of a conveyor belt. Our focus is on disruptions that require using 5-axis machines for production. By omitting the routing before and after the 5-axis machines, we can simplify this problem from an extended job shop to a single-machine scheduling problem. We take a dynamic scheduling approach to handle disruptions, using a predictive-reactive scheduling strategy. Using an initial schedule as input, we make adjustments (rescheduling) after disruptions occur. We chose this approach because disruptions in this research are unpredictable, and defining a stochastic process is not feasible. Rescheduling is thus key to adjusting the schedule based on real-time events.

Next, we determine the rescheduling trigger, choosing between event-driven, periodic or hybrid. A need for periodic rescheduling is eliminated by the use of the model proposed by Van Boxel (2024), and therefore, we select an event-driven strategy. From our literature research, we find that complete rescheduling generally will give the best results, but comes with a long runtime. In the case of event-driven rescheduling, this is not feasible due to the requirement of a short runtime. There, we choose an insertion-based local rescheduling strategy with an expanding time window. The insertion method is best-fit, as this optimises the objective of the problem.

Finally, we assess the rescheduling performance. We can conclude that schedule robustness is essential in limiting the need for event-driven rescheduling. Therefore, in the initial schedule, we will include some robustness. In the implementation phase, we need to find a balance between the robustness level of the initial schedule and the need for rescheduling. Additionally, scheduling stability can support our research in the experimental phase. In our research, we define a schedule as stable when the sequence of produc-

tion is unchanged.

In conclusion, based on the literature, we choose an event-driven, insertion-based rescheduling method with an expanding time window to effectively manage disruptions in a flexible machining cell problem. By balancing initial schedule robustness with the need for rescheduling, we aim to enhance scheduling stability while minimising machine switches and order splits, ultimately ensuring a more resilient and efficient production process.

4| Solution Design

In this chapter, we present the design of the proposed solution, outlining the data requirements, assumptions, and methodologies used. This chapter addresses the third and fourth research questions:

> *3. What should a scheduling solution look like to repair a disrupted schedule?*
> *4. How should a scheduling model be designed that can adapt to disruptions within a reasonable computational runtime?*
> *time?*

We begin by explaining the necessary input data in Section 4.1. Next, Section 4.2 discusses the key assumptions that guide the model. In Section 4.3 we present a Mixed Integer Program (MIP) designed to optimise scheduling. Then, in Section 4.4, we develop a repair heuristic based on the MIP, which is further refined into a comprehensive rescheduling process in Section 4.5.

4.1 Input data

This section elaborates on the input data required for our model.

- Initial schedule
  Our model starts with an initial schedule generated using the solution by Van Boxel (2024), as explained in Section 2.4. As noted in Section 1.2.2, this model is unsuitable for rescheduling due to its long computational runtime. However, it is feasible to use it for generating an initial schedule, since this can be computed in advance while production continues based on the existing schedule. Once the new initial schedule is ready, operators can switch over. The schedule is a list of operations with corresponding starting and ending times of the setup, labour and processing per separate operation. Therefore, one row corresponds to a specific operation of a job. The order in this list allows us to assign an index to each operation, which we use in Section 5.1.2. Using this schedule reduces the planners' workload while maintaining a high-quality baseline.

- Emergency order information
  If an emergency order comes in, we require information on the type of job and its characteristics. This information includes the item number, due date, amount, work order (WO), WO line and item description. Table 4.1 shows an example of the required input.

- Reason for rescheduling
  The cause of disruption affects the model's constraints and optimisation. As a result, the initial time window is determined by the reason for rescheduling, which we include as an input in our model. There are five reasons for rescheduling, detailed in Section 2.1:

  1. Machine failure
  2. Operator illness
  3. Emergency order
  4. Execution delay
  5. Material unavailability

- Start and duration of the disruption
  To ensure that the model can correctly identify what part of the initial schedule was finished and from what moment on the machine can produce again, the duration of the disruption is needed. How these times are obtained is explained in Section 2.1.

Figure 4.1 illustrates how our model interfaces with the solution approach of Van Boxel (2024). In the diagram, blue represents inputs, green indicates outputs, and yellow denotes outputs from the approach

Item	WO	WO Line	Description	Quantity	Due date	Setup time	Labour time	Run time
7123	2364	1	Component A	47	27-05-2025	2 [hrs]	5 [min]	25 [min]

Table 4.1: Example of the input required for an emergency order.

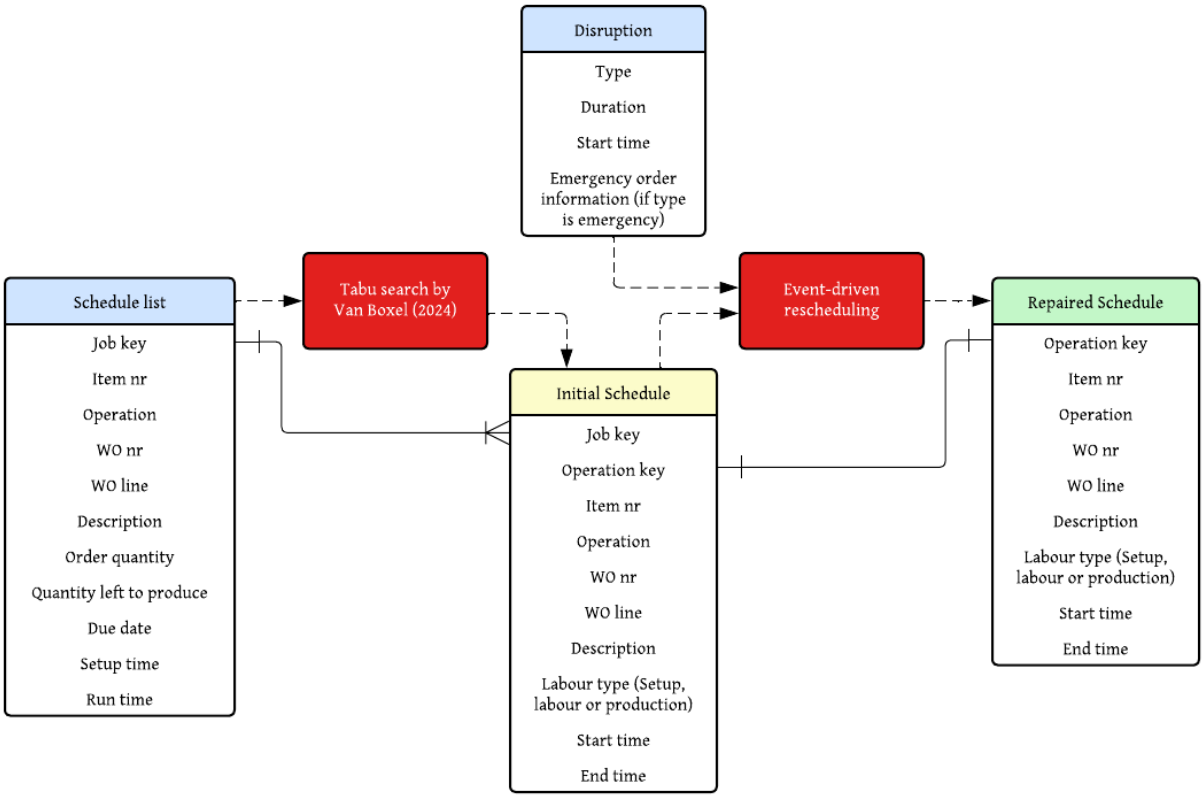


Figure 4.1: The interaction between input and output of the approaches by Van Boxel (2024) and us

by Van Boxel (2024) that serve as inputs for our model. The figure also highlights the relationship between the schedule list and the initial schedule. In the schedule list, each row corresponds to an entire order and may contain a larger quantity. In contrast, the initial and repaired schedules describe operations individually, so multiple rows may relate to a single job. This distinction is made using a unique job key assigned to each operation.

4.2 Assumptions

In this section, we describe the needed assumptions for our model.

- Deterministic processing, labour and setup times
  Processing, labour, and setup times are fixed and do not vary. Therefore, we can assume that these times are deterministic.

- Materials are available
  We assume that materials are always available for production orders. Material shortages are treated as disruptions. Therefore, in the model, we can omit material availability constraints and only adapt release dates for rescheduling in case of such a disruption.

- Shift times
  The machines can only be loaded and unloaded when operators are available. We assume operators

are always available during the week between 7:00 and 16:00, excluding breaks (9:00-9:15, 12:00-12:30 and 14:00-14:15). Weekend shifts are flexible, with a maximum duration of 45 minutes between 9:00 and 15:00. This assumption is valid because unavailability (e.g. due to illness) can be seen as a disruption.

- Dedicated fixtures
  Some job families require dedicated fixtures. Fixtures are job-family specific and cannot be shared between different job families.

- Setup and labour times occur separately from processing times
  Job setup is completed before labour time begins. Furthermore, the finish time of labour time does not have to align directly with the start time of the processing of that job. However, once labour has occurred, that item does occupy a pallet position.

- Tool availability
  In Section 1.2.2, we discussed the tool availability constraint. As it is outside of our scope to connect to the tool database, we follow the assumption used by Van Boxel (2024), treating tools as always available. In moments that a tool is unavailable, we can count this as a disruption due to material unavailability; therefore, enabling rescheduling to generate feasible schedules.

4.3 Mixed-Integer Program

In our research problem, we assume we have a fixed schedule. In this problem, disruptions lead to the infeasibility of the original schedule. We consider disruptions classified into five reasons, as discussed in Section 4.1. To answer the third research question, we formulate an MIP to use for rescheduling the initial schedule after such disruptions. We adapt the mathematical model of Van Boxel (2024), which is a static single-machine FMC, to fit with rescheduling. The main adaptation lies in the set of operations to reschedule. We define this as the set of operations which fall within a time window, whereas Van Boxel (2024) defines this as all operations on the schedule list. Furthermore, we have a difference in objectives. This model aims to minimise two objectives: (1) schedule stability S, defined as the number of jobs that are moved in the sequence of production compared to the initial schedule (see Section 3.3.3), and (2) average tardiness, defined as the average number of days by which products are delivered late. Therefore, we have a multi-objective model. Our MIP is based on expanding time windows, as explained in Section 3.3.2. This is needed for the solution approach and is further explained in Section 4.4.

4.3.1 Notation

We define the set of operations to reschedule, $O_{[l_0,u_m]}$, as the operations that fall within the initial time window $[l_0, u_m]$, determined by the disruption reason. This includes all operations that are planned completely or partially within this time window, either with setup, labour, or processing. The initial time window is the minimum number of operations that require rescheduling due to the disruption, resulting in the shortest computational runtime required to reconstruct a feasible schedule. Then, we expand this time window and therefore regenerate the schedule with more operations included in rescheduling, resulting in better solutions. The time window expands in each iteration m, until a maximum computational runtime of τ^{max} or the maximum number of expansions M has been reached.

A job j may consist of multiple operations O_{ij}, all of which are performed on the same machine. Each operation requires manual setup and labour before processing, with a given setup time S_{ij} and labour time L_{ij}. The setup of an operation entails preparing the machine by putting in the correct parameters. This only occurs once per operation, so if multiple products of the same item are made, the setup only has to be done once. The labour time entails fixing products on pallets and putting them in the machine. These tasks are performed by an operator on the machine. Once setup and labour are complete, the machine begins processing the job, with a processing time P_{ij}. Switching between jobs within the machines happens automatically and only requires a set pallet changing time. Each job has a given release date R_j and

due date D_j.

The complete list of notation we use is as follows:

Subscripts

i	operation index
j	job index
p	pallet index
m	time window index

Sets

$J_{[l_0,u_m]}$	All jobs (partially) within time window $[l_0, u_m]$
$O_{[l_0,u_m]}$	Set of all operations (partially) within time window $[l_0, u_m]$
O_{ij}	Operation i of job j
P	Set of pallets
$Q_{[l_0,u_m]}$	Set of all production orders within time window $[l_0, u_m]$
F_{ij}	Setup family of operation i of job j

Parameters

OB^{week}	Operator begin time during the week
OE^{week}	Operator end time during the week
$OB^{weekend}$	Operator begin time during the weekend
$OE^{weekend}$	Operator end time during the weekend
ω	Objective weights
N_j	Number of operations needed on job j
P_{ij}	Processing time of operation i of job j
S_{ij}	Setup time of operation i of job j
L_{ij}	Labour processing time of operation i of job j
D_j	Due date of job j
R_j	Release date of job j
W	Change time of pallets
DF_{ij}	Dedicated fixtures for operation i of job j
H	Large positive number
M	Maximum number of time window expansions
c	The first job that should be rescheduled
Δu_m	The value with which the upper bound increases at iteration m
(l_0, u_0)	Initial time window with lower bound l_0 and upper bound u_o

Decision variables

x_{ijp}	Binary variable, 1 if operation i of job j is planned on pallet p, 0 otherwise
$y^{old}_{iji'j'}$	Binary variable, 1 if operation O_{ij} is planned before $O_{i'j'}$ *before* rescheduling
$y^{new}_{iji'j'}$	Binary variable, 1 if operation O_{ij} is planned before $O_{i'j'}$ *after* rescheduling
$z_{iji'j'p}$	Binary variable, 1 if operation O_{ij} is planned before $O_{i'j'}$ on pallet p
a_{ij}	Binary variable, 1 if operation O_{ij} is rescheduled
s^{start}_{ij}	Start setup time of operation O_{ij}
s^{end}_{ij}	End setup time of operation O_{ij}
l^{start}_{ij}	Start labour time of operation O_{ij}

l_{ij}^{end} End labour time of operation O_{ij}
t_{ij}^{start} Starting time of operation O_{ij}
t_{ij}^{end} Ending time of operation O_{ij}
(l_0, u_m) Extended time window with variable upper bound u_m
S The total number of jobs moved
T_{ij} Tardiness of operation O_{ij}

4.3.2 Objective function

The objective function, which can be found in Equation 4.1, consists of two parts. The first part is the instability of the schedule, defined by the number of changes in the sequence of production, multiplied by weight ω. We use this measure in the objective function, as modifications to the schedule of one machine can propagate through the internal supply chain. The interdependence of machines causes scheduling complexity, and thus, schedule stability is important. The second part is on the average tardiness of the schedule, multiplied by $1-\omega$. This is calculated by adding the tardiness of all operations and dividing this by the total number of operations. We use the regenerated schedule of the coming week to calculate these values instead of only the rescheduled time window, such that we can compare the initial schedule with the repaired schedule.

$$\text{Min } \omega * S + (1-\omega) * \left(\frac{1}{|O|} * \sum_{j}^{J} \sum_{i}^{N_j} \max\left(0, C_{ij} - D_j\right) \right) \tag{4.1}$$

4.3.3 Constraints

The model is subject to a series of constraints, which are elaborated upon in this subsection.

$$\sum_{p}^{P} x_{ijp} = 1 \qquad \forall i,j \tag{4.2}$$

$$\sum_{j}^{J \setminus \{j\}} y_{iji'j'}^{old} = 1 \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.3}$$

$$\sum_{j}^{J \setminus \{j\}} y_{iji'j'}^{new} = 1 \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.4}$$

$$z_{iji'j'p} + z_{i'j'ijp} = x_{ijp} * x_{i'j'p} \qquad \forall p \in P, \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.5}$$

Constraint 4.2 ensures that all jobs and corresponding operations are planned. Then, Constraints 4.3 and 4.4 ensure that all jobs have a preceding job, before and after rescheduling. Constraint 4.5 ensures precedence within pallets, consequently also restricting the planning of a maximum of one job at a time per pallet.

$$s_{ij}^{start} \geq R_j \qquad \forall i,j \tag{4.6}$$

$$s_{ij}^{end} = s_{ij}^{start} + S_{ij} \qquad \forall i,j \tag{4.7}$$

$$l_{ij}^{start} \geq s_{ij}^{end} \qquad \forall i,j \tag{4.8}$$

$$l_{ij}^{end} = l_{ij}^{start} + L_{ij} \qquad \forall i,j \tag{4.9}$$

$$t_{ij}^{start} \geq l_{ij}^{end} \qquad \forall i,j \tag{4.10}$$

$$t_{ij}^{end} = t_{ij}^{start} + P_{ij} \qquad \forall i,j \tag{4.11}$$

The following set of constraints sets the start and ending times for setup, labour and processing for each operation. First, Constraint 4.6 ensures that all jobs are only started once the release date of the job is

reached. Constraint 4.7 sets the ending time for setting up an operation. Constraint 4.8 sets the starting time of labour based on the ending time of setting up an operation. Constraint 4.9 sets the ending time of labour, and Constraint 4.10 sets the processing starting time based on this. Lastly, Constraint 4.11 sets the ending time of an operation.

$$s_{ij}^{end} = s_{i'j'}^{end} \qquad \forall i,j: \quad F_{ij} = F_{i'j'} \tag{4.12}$$

$$t_{i'j'}^{end} \geq t_{ij}^{end} + P_{i'j'} + W - (1 - Y_{iji'j'}) * H \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.13}$$

$$l_{i'j'}^{start} \geq t_{ij}^{end} - (1 - z_{iji'j'p}) * H \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.14}$$

$$t_{ij}^{end} \leq l_{i+1,j}^{start} \qquad \forall j, i = 1, ..., (N_j - 1), N_j > 1 \tag{4.15}$$

$$l_{i'j'}^{start} \geq l_{ij}^{end} \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.16}$$

Next, we define the starting and ending times of each variable for preceding jobs. Constraint 4.12 is regarding the setup time for a family of jobs. This is always zero, and therefore, the ending of setup times for preceding operations within the same family is equal. Constraint 4.13 sets the ending time of a following job based on the ending time of its predecessor, its processing time and the required pallet change time. Constraint 4.14 ensures that an operation only starts labour once its preceding operation on the same pallet is finished with processing. Constraint 4.15 ensures that the required labour for an operation is only started once the preceding operation of the same job is finished. Lastly, Constraint 4.16 ensures that the labour for an operation is finished before starting labour for the next job.

$$\sum_{p}^{P} x_{ijp} \leq DF_{ij} \qquad \forall i,j \tag{4.17}$$

Constraint 4.17 ensures that the total amount of pallets a product can be planned on does not exceed its corresponding fixture availability.

$$OB^{week} \leq l_{ij}^{start} \bmod 24 \leq OE^{week} - L_{ij} \qquad \forall i,j: \left\lfloor \frac{l_{ij}^{start}}{24} \bmod 7 \right\rfloor < 5 \tag{4.18}$$

$$OB^{weekend} \leq l_{ij}^{start} \bmod 24 \leq OE^{weekend} - L_{ij} \qquad \forall i,j: \left\lfloor \frac{l_{ij}^{start}}{24} \bmod 7 \right\rfloor > 5 \tag{4.19}$$

$$OB^{week} \leq s_{ij}^{start} \bmod 24 \leq OE^{week} - S_{ij} \qquad \forall i,j \tag{4.20}$$

$$\left\lfloor \frac{s_{ij}^{start}}{24} \bmod 7 \right\rfloor < 5 \qquad \forall i,j \tag{4.21}$$

This set of constraints ensures that we adhere to operator availability. First, Constraint 4.18 ensures that labour is only started during working hours on weekdays, whereas 4.19 does the same for the weekend. Constraint 4.20 ensures that setup is done during operator availability in the week, Constraint 4.21 restricts the model from having setup starting in the weekend.

$$u_m = u_{m-1} + \Delta u_m \qquad \forall m \setminus \{0\} \tag{4.22}$$

In this set of constraints, we define the time window. Constraint 4.22 defines the time window upper bound based on the preceding bounds, therefore impacting the set of jobs to reschedule, $J_{[l_0,u_m]}$. All jobs that are either completely or partially planned within that time window in the initial schedule are part of $J_{[l_0,u_m]}$. The parameter with which this bound changes, Δu_m, is determined based on either a linear, exponential or logarithmic expansion scheme. In Section 5.3 we show these equations and perform experiments to determine which scheme should be used.

$$a_{ij} \geq y^{new}_{iji'j'} + y^{old}_{iji'j'} \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.23}$$

$$a_{ij} \geq y^{old}_{iji'j'} + y^{new}_{iji'j'} \qquad \forall O_{ij}, O_{i'j'} \in O: \quad O_{ij} \neq O_{i'j'} \tag{4.24}$$

$$S = \sum_{j}^{J} \sum_{i}^{I} a_{ij} \tag{4.25}$$

$$t^{end}_{ij} \leq D_j + T_{ij} \qquad \forall i, j, \tag{4.26}$$

These constraints set the variables used for the objective function. Constraints 4.23 and 4.24 set the value of a_{ij} to 1 if the sequence of operations is different after rescheduling. Constraint 4.25 then sets the total schedule stability. Constraint 4.26 sets the tardiness of each job based on its ending time and due date.

$$x_{ijp}, y^{old}_{iji'j'}, y^{new}_{iji'j'}, z_{iji'j'p}, w_{iji'j'p}, a_{ij} \in \mathbb{B} \qquad \forall i, j, p \tag{4.27}$$

$$s^{start}_{ij}, s^{end}_{ij}, l^{start}_{ij}, l^{end}_{ij}, t^{start}_{ij}, t^{end}_{ij}, T_{ij} \geq 0 \qquad \forall i, j \tag{4.28}$$

$$l_0, u_m \geq 0 \qquad \forall m \tag{4.29}$$

Lastly, we have the sign constraints for all variables (Constraints 4.27-4.29).

4.4 Solution approach

In the previous section, we already determined that we use the model of Van Boxel (2024) to generate an initial schedule. If any disruptions occur, we require a rescheduling approach. The MIP of the previous section provides us with a basis for this model, showing the objective and all required constraints. A possible solution approach would be to solve the MIP for the remaining period until a new initial schedule is generated. However, in Section 2.5, we discuss that HTM Aerotec has a computational runtime limit of a few minutes. Hence, we cannot use the MIP as it will take too long. Therefore, to answer the fourth research question, taking into account computational runtime, we use a repair heuristic based on a best-fit insertion and expanding time windows to solve our rescheduling problem.

The repair heuristic is organised into three hierarchical stages. The Main Algorithm oversees the overall rescheduling process and identifies the set of jobs requiring rescheduling. It first delegates to the Repair Sub-procedure, which updates the schedule based on the placements determined by the Best Fit Procedure. The best-fit procedure computes the optimal insertion points, the points in the sequence of operations, for all affected operations within the operation sequence. Once the repair sub-procedure is complete, the control returns to the main algorithm, which may expand the time window if the maximum computational runtime or number of iterations has not yet been reached. The time window defines the jobs which are rescheduled; all jobs which (partially) fall within this window are included in rescheduling. Therefore, expansion of this time window results in a larger set of jobs to reschedule. More expansions can thus potentially lead to better solutions, as this provides more flexibility in optimising the schedule for the changed situation. In the next section, we first discuss the computational runtime. Then, in the following sections, we will provide a detailed description of the individual algorithms and their respective functions within the rescheduling process.

4.4.1 Computational runtime

First, we discuss the computational runtime in more detail. At HTM Aerotec, the problem instances are very large, sometimes containing over 4000 operations to be performed by one machine over ten months; about 200 operations per week. These large problem instances result in long computational runtimes depending on how many operations are rescheduled. However, the number of operations within a certain period and the total length of the planned period vary largely over time. Therefore, determining computational runtimes and the number of operations to reschedule is not straightforward. For this, first, we consider the time horizon we are rescheduling. The planning department updates the initial schedule weekly

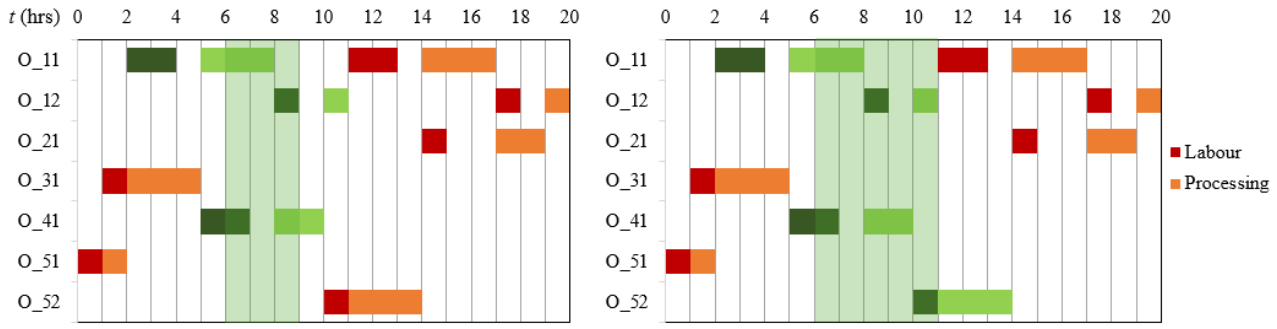


Figure 4.2: An example of a time window and an expansion for a disruption from $t = 6 - 8$

if significant disruptions occur. Consequently, rescheduling beyond one week is unnecessary, as the planning department will intervene with a new schedule if further disruptions arise. Thus, the rescheduling algorithm only needs to adjust operations within a 168-hour window from the moment of disruption, ensuring that the schedule remains feasible and aligned with real-world planning cycles. In the algorithms, we calculate this as the total rescheduled production hours P_{total}. To further minimise the computational runtime, the number of operations to reschedule should be minimised. Contrarily, to accurately calculate the effect and best solution, the number of operations to reschedule should be maximised. To effectively balance this trade-off, we chose to use expanding time windows. This approach allows us to control the computational effort required to solve our problem instance.

4.4.2 Main algorithm

The Main Algorithm presents the pseudocode for our main rescheduling procedure. This algorithm follows the notation introduced in Section 4.3.1. Here, M represents the maximum number of allowed iterations (or time window expansions), and the parameters l_o and u_0 define the initial lower and upper bounds of the time window. These parameters correspond with the start and end of the disruption as explained in Section 2.1. The left graph in Figure 4.2 shows an example of such an initial time window. The algorithm begins by extracting all relevant information regarding the disruption, such as its type, duration, and the first job that must be rescheduled. Subsequently, the algorithm determines the disruption start time based on the first affected job. The next line calculates the initial time window based on the disruption type. In the case of an emergency order, we have an initial time window with size zero, where the only job to reschedule is the emergency order.

Algorithm 1 Main Algorithm

```
procedure MainRepairAlgo(disruption information, operator hours)
    Extract key parameters from disruption information
    Set disruption_time ← start of disrupted operation
    Initialise [l_0, u_0] based on disruption type
    while runtime < max runtime and iterations < M do
        Identify operations to repair in the current time window
        if Disruption type == emergency job then
            Append all operations of emergency job to operations to repair
        Reschedule using (Repair Sub-procedure)
        Expand time window bounds ([l_0, u_m])
    Compute total and maximum tardiness
    Compute objective (Equation 4.1)
    return tardiness, objective value and repaired schedule
```

With this information, the code enters a `while` loop. In each iteration, the algorithm identifies all operations that fall within the time window and that thus require repair. In Figure 4.2, these jobs are green. It then adds the emergency job if the disruption type is an emergency order. With this list of operations, the schedule is repaired using the Repair Sub-procedure. This results in a schedule with a time horizon of 168 hours. Then, the time window is expanded for the next iteration. An example of such an expanded time window and its corresponding jobs is shown in Figure 4.2. In the case of an emergency order, the first expansion is the expansion of one hour, after which the normal expansion methods are used. This is needed due to the initial window size of zero in those cases. This `while` loop continues until either the maximum runtime or the maximum number of iterations M is reached. Once the loop terminates, the total tardiness of all operations and the maximum tardiness are calculated, and the objective value is computed. These results, together with the repaired schedule, are the output of the algorithm.

Figure 4.3 provides a visual example of one iteration of this algorithm. This example is a small and simplified version of the problem instances at HTM Aerotec, where the number of operations daily can be up to ten times as large and more constraints are present (e.g. manual labour required for each operation, dedicated fixtures, etc.). In the original schedule, we have five items with a total of nine operations to be completed over one day. Each row represents a specific operation, and the horizontal axis shows the timeline from hour 1 to 24. The green lines show the insertion positions, meaning where in the sequence of operations an operation could be inserted. These positions are determined taking into account the sequence of all other operations, which are the red blocks. Thus, the time at which they start is not important for the insertion point. At hours 1 and 2, the machine is unavailable, as indicated by the black disruption blocks. However, in the original schedule, the first operation of job 1 is planned during this disruption (shown as the striped "disrupted" block). This makes the original schedule infeasible and requires rescheduling of that operation. The rescheduled operation can be placed anywhere in the sequence. However, if it is placed after its dependent operation, e.g. second operation of the same job, we also need to reschedule that operation to maintain schedule feasibility. The insertion positions are then determined again using the red blocks, but now starting after the first operation. Hence, if the first operation is rescheduled at position 3 and the second operation is scheduled immediately after, that would be insertion position 1 for the second operation. Below, we discuss three possible solutions and their implications, of which the Best-Fit algorithm would choose the best option:

1. The disrupted operation is inserted at the first position of the sequence, immediately after the disruption ends (at hour 3), as indicated by the orange "operation to reschedule" block for the rescheduled operation. This pushes all subsequent operations backwards in time. The advantage of this solution is that we only reschedule one operation, resulting in maximised stability of the overall schedule. However, other operations are delayed, which could result in tardiness if we violate due dates.

2. Here, the disrupted operation is inserted into the second position in the sequence (now starting at hour 6). However, because its second operation was originally scheduled first, it must be moved as well to maintain the correct order. We insert the second operation at the second position (starting at hour 13), both shown in orange. This solution moves two operations, decreasing the schedule stability compared to the first solution. However, this solution may result in lower tardiness (e.g., if item 2 has a due date around hour 9).

3. In this case, the first operation of item 1 is moved to the eighth available position (starting at hour 20), and the second operation is placed immediately afterwards at position 1 (hour 23), maintaining the precedence constraint. This solution has the same stability as the second solution. However, depending on the due dates of the other items, this solution might result in lower tardiness.

The right graph in Figure 4.2 shows an expanded time window, based on the initial time window on the left graph. This expansion is performed using one of four strategies: linear (Equation 4.30), exponential (Equation 4.31), logarithmic (Equation 4.32), or percentage based (Equation 4.33). At the end of each iteration, these equations are used to determine the step size with which we increase the upper bound of the time window. This results in an increasing size of time windows each iteration. Figure 4.4 shows an example of

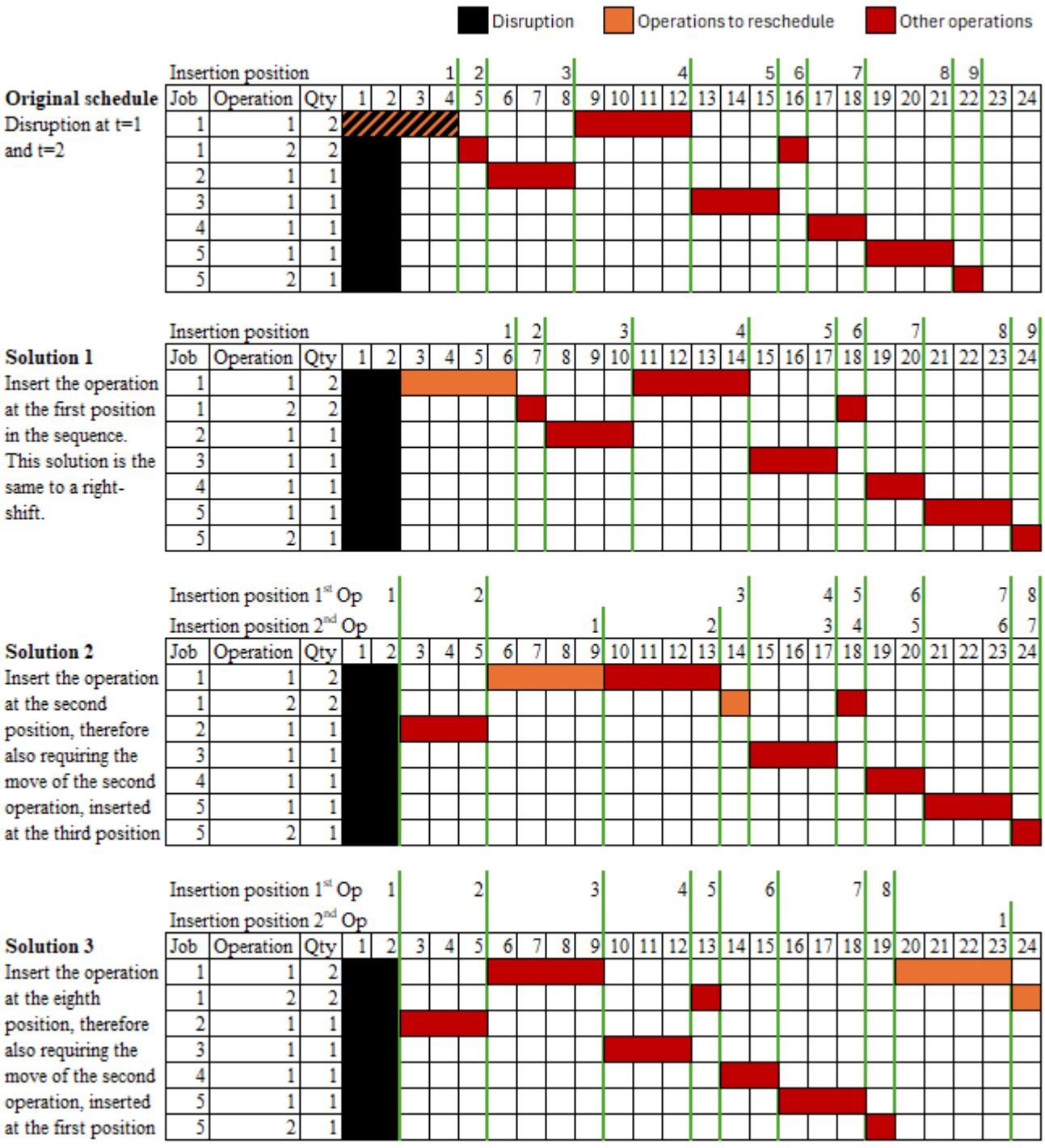


Figure 4.3: Example of a disrupted schedule due to machine failure with three possible repaired schedules. t is expressed in hours.

how the size of the time window is affected over time per method. There, we can see a larger expansion at the start for the logarithmic method, while the percentage-based method starts with the smallest window and ends with the largest. The corresponding equations determine the step size with which the upper bound u_m increases. In these equations, t_h is the total time horizon and m is the current iteration. For the expansion, only one of these three strategies is used. In Section 5.3, we perform experiments to find the most suitable expansion method.

$$\Delta u_m^{linear} = \frac{t_h - u_0}{M} \qquad \forall m, c \quad (4.30)$$

$$\Delta u_m^{exponential} = m^{\frac{\log(t_h - u_0)}{\log(M)}} \qquad \forall m, c \quad (4.31)$$

$$\Delta u_m^{logarithmic} = \log(m) * \frac{t_h - u_0}{\log(M)} \qquad \forall m, c \quad (4.32)$$

$$\Delta u_m^{percentage} = b * (u_m - u_o) \qquad \forall m \quad (4.33)$$

4.4.3 Repair algorithm

In the Main Algorithm, we use the Repair Sub-procedure to reschedule the operations in $O_{[l_0,u_m]}$ and update all corresponding start and end times. This repair algorithm starts with a part of the initial schedule, which is up to the point where the schedule is disrupted. Then, iteratively, one by one, operations are added in a determined sequence, calculating corresponding processing times. We will call this partial solution the working solution. The Repair Sub-procedure begins by calling the Best Fit Procedure, which determines the optimal insertion positions for each operation in $O_{[l_0,u_m]}$. These insertion positions and the sequence we will use in this algorithm are determined based on the order in which the machine produces. As an example, in Figure 4.3 in the original schedule, the first in the sequence is the disrupted operation, then the second operation of item 1, then the operation of item two and so on. The insertion points are marked with green lines, which are between every operation. The Best-Fit algorithm chooses insertion positions for all operations in $O_{[l_0,u_m]}$ and, based on these insertion positions, this repair algorithm determines the new sequence and the corresponding production times.

After retrieving all insertion positions, the algorithm iterates over all affected operations in $O_{[l_0,u_m]}$. For each operation, it retrieves the position to which it should be moved. Next, the algorithm identifies up to what position the working solution has been updated. Then, a loop begins to bridge the gap between

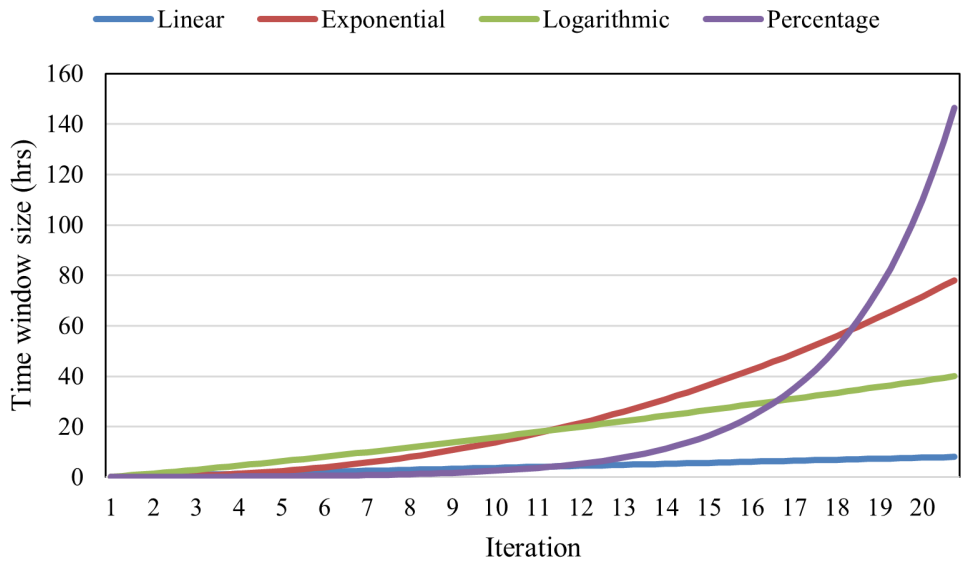


Figure 4.4: An example of the size of the time window over time per method

the working solution and the position in which the current operation should be added. In other words, if the working solution is not yet updated up to the insertion point of the current operation, all preceding operations of the sequence should be added first. In this loop, the algorithm sequentially adds operations from the sequence using the UpdateSolution procedure. This loop terminates either when the required insertion position is reached or when the working schedule reaches the maximum time horizon of one week (168 hours). Once the schedule is updated up to the insertion position, the operation requiring rescheduling is added to the solution. Again, if the time horizon has reached 168 hours, the algorithm terminates. Once all operations in $O_{[l_0,u_m]}$ are rescheduled, the algorithm checks if any operations from the initial sequence can still be added. If the working solution is still within the time horizon, a final loop adds the remaining operations sequentially until either all operations are added or the time horizon maximum has been reached. Finally, the algorithm returns the updated solution.

Algorithm 2 Repair Sub-procedure

```
1:  procedure RepairAlgorithm(operations_to_plan, operator hours)
2:      operations_to_plan ← Calculate_Best_Fit(operations_to_plan)
3:      for operation in operations_to_plan do
4:          position ← best-fit position for operation
5:          last_updated ← current end of updated schedule
6:          for operation_j = last_updated + 1 to position do
7:              Extract operation_j from initial solution
8:              UpdateSolution(operation_j)
9:              if current schedule time horizon ≥ 168 then
10:                 return solution
11:         UpdateSolution(operation)
12:         if current schedule time horizon ≥ 168 then
13:             return solution
14:     last_original ← final position in initial solution
15:     last_current ← final position in current solution
16:     if last_original > last_current and current schedule time horizon < 168 then
17:         for operation_j = last_current + 1 to last_original do
18:             Extract operation_j from initial solution
19:             UpdateSolution(operation_j)
20:             if current schedule time horizon ≥ 168 then
21:                 return solution
22:     return solution
```

4.4.4 Best-Fit algorithm

The final procedure, Best Fit Procedure, determines the optimal position in the sequence for rescheduling an operation. As discussed in Section 4.4, several strategies exist for determining this position. Among these, we adopt a best-fit method, which aims to minimise the objective function (Equation 4.1) by selecting the position that offers the best trade-off between schedule stability and tardiness, while considering machine constraints. The algorithm first calculates the best insertion position for all jobs to reschedule. This results in a new sequence of jobs, which is used to construct the new production schedule. Therefore, our algorithm is considered to be greedy. As the objective function (Equation 4.1) is a minimisation problem, here also the minimal score is the optimal score.

One important constraint to consider in the best-fit algorithm is the precedence constraint. Some jobs require multiple operations in a pre-determined order. If an operation is rescheduled, we always need to ensure that its successors are scheduled later. If the operation is rescheduled before its successors, we do not require rescheduling of the successors. Hence, the set of jobs to reschedule can be expanded with successors based on the insertion position of a preceding operation. We call these successors dependencies.

Algorithm 3 Best Fit Procedure

```
1:  procedure CALCULATE_BEST_FIT(operations_to_plan)
2:      for operation in operations_to_plan do
3:          for position = operation_index to next 50 positions do
4:              if operation has subsequent operations then
5:                  Find dependent operations before position
6:                  Add to dependencies, increment nr_jobs_to_move
7:              Calculate tardiness at current position
8:              calculate score
9:              Record score, position, and dependencies in score_records
10:     for operation in operations_to_plan do
11:         operation_scores ← scores for this operation from score_records
12:         Choose the entry with the minimum score
13:         Extract best_location, dependencies
14:         Add best_location to operation_moves
15:         for dependant_operation in dependencies do
16:             Set dependent_operation position on the best position after best_location
17:     for operation in operation_moves do
18:         Update operations_to_plan with new location
19:     Sort operations_to_plan by location
20:     return operations_to_plan
```

The Best Fit Procedure consists of three main loops. The first loop iterates over each operation in $O_{[l_0,u_m]}$. For each operation, the algorithm evaluates the next hundred positions in the sequence at most. At HTM Aerotec, up to 200 operations each week are reached. We limit the algorithm to the next hundred positions, as moving an operation further back in the schedule leads to excessive tardiness. Additionally, this further decreases the computational runtime of our algorithm. At each potential position, it identifies any dependent operations that must be moved due to precedence constraints; these are stored as *dependencies*. The number of required moves and the expected tardiness at that position are used to calculate a score. To find the true best fit for each position, the entire schedule should be generated, and the total tardiness and stability should be calculated. However, generating a schedule takes a long computational runtime. Therefore, we use estimations to calculate the score. Estimations of the tardiness and stability can lead to suboptimal insertion positions, especially taking into account constraints such as shift times. Therefore, we have three other scoring methods which are adapted based on the constraints of our scheduling problem. The four scoring methods are:

1. **Tardiness and moves**
   This method reflects the objective function, balancing two goals: minimising tardiness (T_{ij}) and limiting the number of moves (a_{ij}). The parameter ω determines their relative importance. This method estimates the tardiness of the next fifty jobs by dropping all constraints except shift hours. Then, the scoring formula is defined as:

$$score^{tardiness} = \omega * a_{ij} + (1-\omega) * T_{ij} \tag{4.34}$$

2. **Similar timeslot**
   This method assumes that the original scheduling time is near-optimal and prioritises rescheduling to a similar time. It accounts for differences between weekdays and weekends, which vary in available manual labour hours. The score increases with greater deviation in labour or production times, in three cases:

   - Both the operation and position fall on weekdays or both on weekends: compare only labour times.

- One is scheduled on the weekend and the other at night: compare labour and production times.
- All other combinations: weigh labour time differences more heavily, with production time as a secondary factor.

Variables: h_i^L and h_p^L are the hours at which labour for the operation and position is scheduled, respectively. h_i^P and h_p^P refer to their production hours. d_i^L and d_p^L are the corresponding weekdays.

$$score^{same\_time} = \begin{cases} |h_i^L - h_p^L| & \text{if } (d_i^L < 6 \wedge d_p^L < 6) \vee (d_i^L \geq 6 \wedge d_p^L \geq 6) \\ |h_i^L - h_p^L| + |h_i^P - h_p^P| & \text{if } (d_i^L \geq 6 \wedge h_i^P > \text{OE}^{week}) \vee (d_p^L \geq 6 \wedge h_p^P > \text{OE}^{week}) \\ 2|h_i^L - h_p^L| + |h_i^P - h_p^P| & \text{otherwise} \end{cases} \tag{4.35}$$

3. **Ratio of hours left**
   This method compares the manual-to-automatic labour ratio r_i of the operation with the remaining ratio at the target position r_p until the start of the next shift. The goal is to align the characteristics of the job with the availability of remaining resources, such that we can fully utilise the production hours of the machines:
   $$score^{ratio\_left} = |r_i - r_p| \tag{4.36}$$
4. **Weekly ratio balance**
   This final method evaluates whether the operation's manual/automatic labour ratio r_i is higher than the overall weekly average r_r. If so, the algorithm prefers a position with a lower ratio r_p; otherwise, it prefers a higher one. This encourages balancing labour usage across the week:
   $$score^{ratio\_balanced} = \begin{cases} r_p & \text{if } r_i > r_r \\ 1 - r_p & \text{otherwise} \end{cases} \tag{4.37}$$

In the second loop, the algorithm again iterates over all operations in $O_{[l_0,u_m]}$. For each operation, it selects the recorded position with the minimum score. This is the best-fit location for that operation. All dependent operations are assigned a position immediately after the main operation to maintain precedence. The third and final loop updates the array $O_{[l_0,u_m]}$ with the new position assignments, including dependencies. The result is sorted by position. The final output is a dictionary, which will be used in the repair heuristic.

4.5 Proposed Rescheduling Process

The initial (re)scheduling process, illustrated in Figure 2.1 and Figure 2.2, puts a lot of manual labour on the planning department. By implementing a model to generate updated schedules automatically, this manual effort can be substantially reduced. However, the integration of this model requires adaptation to the (re)scheduling process. In the initial process, the initial schedule was manually constructed and adapted once new orders arrived. We eliminate the manual labour by using the model proposed by Van Boxel (2024) to generate an initial schedule. This model should be used to generate a new schedule bi-weekly, such that disruptions always occur in the early stages of the planning horizon. This maximises the efficiency of the rescheduling model. Once the initial schedule becomes infeasible due to any disruption, the rescheduling model should be used. This way, a feasible schedule is available within a reasonable time.

Using the proposed model requires changes in the rescheduling process. Figure 4.5 shows the proposed rescheduling process. In the figure, we can see that all involved parties can be the first to notice a disruption within the process. If an operator notices this, he can immediately verify the type of disruption according to Section 4.1, and use the rescheduling algorithm at the machine. The operators are the ones to use the algorithm, as they have all the input information and are the ones who require a quick solution to ensure that the machine utilisation stays at the aimed level. Therefore, they can immediately update

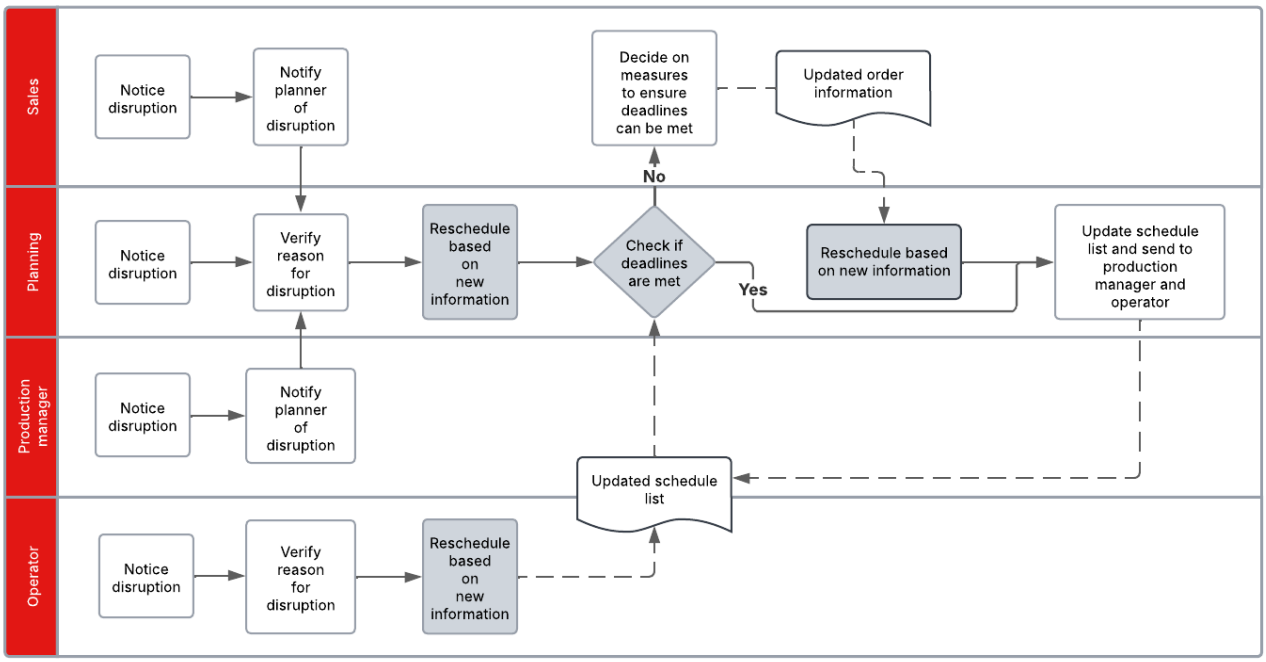


Figure 4.5: The proposed rescheduling process

the schedule list and start with the production of other parts. This way, the downtime in those cases is limited, as production continues while the planning department checks the planning for due date feasibility. After starting with the new schedule list, the operator informs the production manager, who in turn informs the planner. If one of the other involved parties notices the disruption, they notify the planner first. The planner can then use and check the newly found schedule list, especially for meeting due dates. If the planning is accepted, they send this to the operator. However, if due dates are not met, they discuss the implications with the sales department. Based on this, they decide if the newly found planning is sufficient or if other actions should be taken. Such actions include splitting up an order or pushing back the due dates of certain customers. With this decision, the sales department can update the deadlines and release dates. Then, the planner updates the schedule list and sends this to the operator.

4.6 Conclusion

In this chapter, we have addressed the third and fourth research questions by developing a scheduling solution that can respond effectively to disruptions. Initially, we proposed an MIP that identifies and reschedules operations within a defined time window. This model minimises job movement and tardiness while satisfying all operational constraints, such as operator availability. To ensure compatibility with the algorithm and the production process, we adjusted the input data structure.

While the MIP provides a theoretical foundation, it is impractical for real-time implementation as the large number of constraints results in high computational demands. Consequently, we developed a rescheduling approach inspired by the MIP model but adapted using expanding time windows and heuristic repair strategies. We proposed four time window expansion strategies, three of which are adapted from Kuster et al. (2010). Rescheduling is performed using a best-fit insertion algorithm that evaluates positions in the existing schedule based on four scoring strategies: tardiness and stability, similarity of timeslots, the ratio of hours left and the weekly ratio balance. This approach enables fast and effective schedule repair with maximised stability. With this model, we have a scheduling solution and a model able to repair the schedule after disruptions within a reasonable time.

5| Experiments

In this chapter, we aim to answer the fifth sub-question from Section 1.3:

> *What experimental design and validation metrics are required to optimise and evaluate the solution's effectiveness for real-world application?*

In this chapter, we identify and analyse multiple experiments to answer the question. Section 5.4 discusses the weigh-off between a longer runtime and the objective value. In Section 5.3, we discuss the method of expansion for the time windows. Section 5.5 analyses the differences between the performance of the initial and manual schedule versus after rescheduling. Lastly, in Section 5.8, we conclude on the findings of this chapter.

5.1 Problem Instances and Experiments

For all experiments, we use two different initial problem instances, resulting in two initial schedules for experimentation. We choose one instance per machine in our scope. For all experiments except generalisation, the instances of machines 538 and 539 are used. Then, in Section 5.7 we use the other two instances to demonstrate how the proposed models perform on the other 5-axis machines. Due to the large planning horizon, represented by the column 'Time horizon' (showing the number of days left before the last order needs to be completed), we can assume that these instances represent an average mix of item types and characteristics. We use these instances to run the model of Van Boxel (2024), resulting in the initial schedule we require as input, as mentioned in Section 4.1. We use these schedules in combination with the scenarios, explained in Section 5.1.2, to simulate disruptions over the entire time horizon.

Machines 538 and 539 have a significantly higher workload compared to machine 533. Furthermore, machine 538 has many items closer to the current date, and more space later in the time horizon. On the contrary, machine 539 has an evenly spread workload. This is of importance when looking at the objective value. HTM Aerotec accepts orders based on a lower utilisation rate than the utilisation rate of the initial schedule. Therefore, over time, possible tardiness is decreased. This trend is shown in Figure 5.1. That graph shows example output of our model, based on a tardiness and stability best fit method and a linear time window expansion method. The y-axis shows the tardiness of the schedule per scenario, where the x-axis is increasingly sorted based on the initial breakdown moment. These breakdown moments are spread across the entire time horizon, where scenario 27 is a disruption at the 60^{th} operation, while scenario 0 is a disruption at operation 3671. As machine 538 has a slightly higher workload, and especially in the beginning, a higher risk for tardiness, it is likely that the objective value for machine 538 is higher than the value of machine 539. A further difference between these instances lies in the type of jobs, or the job mix, which are included. Different jobs require different amounts of labour, setup and runtime. Therefore, the manual-to-automatic labour ratio, explained in Section 2.5, differs.

5.1.1 Experiment overview

Table 5.1 provides an overview of the experiments conducted in Sections 5.2 up to and including 5.6. Each experiment is conducted independently, varying only one factor at a time (see the column 'Options') while keeping all others constant (column 'Experiment settings'). Based on these experiments, we conclude the optimal best-fit method. Then, with the chosen best-fit method, we experiment and conclude on the expansion method. Lastly, we experiment with the maximum computational runtime using the chosen best-fit and expansion method. Using these insights, we compare the solution approach to the initial schedule and manual schedule to conclude the performance of our approach. Then, we conduct a sensitivity analysis on due dates and job mix to assess how schedule flexibility and variations in manual labour

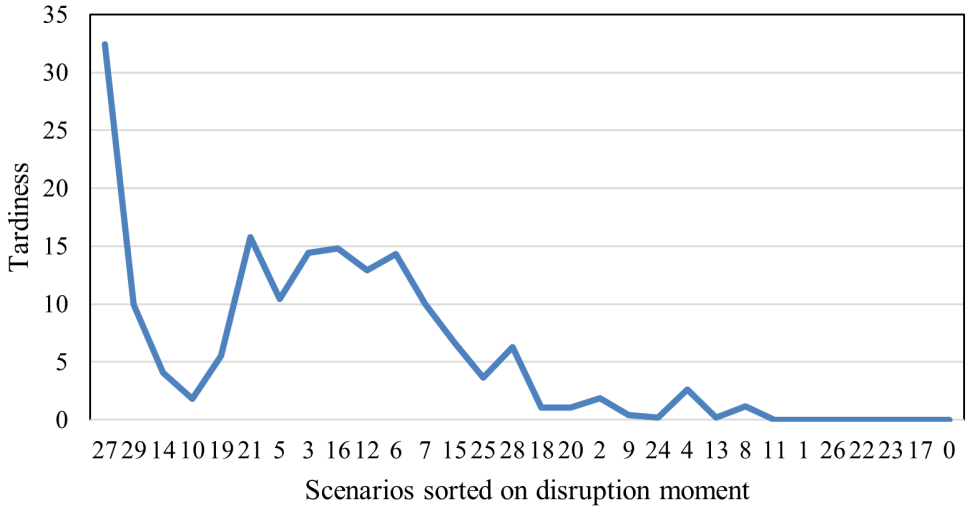


Figure 5.1: Tardiness of the schedule per scenario and method sorted on the initial disruption moment

requirements impact the model's performance. The due date experiments are done by changing the due dates obtained from the input. The normal due dates are the due dates as were set by HTM Aerotec in their planning. To experiment, we set all due dates one or two weeks later, therefore creating more room in the schedule. The job mix experiments are based on the manual-to-automatic labour ratio. By changing that ratio, we can simulate a different job mix. To reach different ratios, we multiply the labour times with a factor between 0 and 1, decreasing the ratio.

5.1.2 Scenario sampling

The rescheduling algorithm has varying performance. This is due to the type of disruption and the timing and original duration of the time window. Hence, it is necessary to assess numerous scenarios. For a realistic approximation of the algorithm's effectiveness, an infinite number of scenarios would be ideal, such that we can evaluate all possible disruptions. However, that is not feasible considering the practically possible runtime. Therefore, we need to find a balance between a large enough number of scenarios and a reasonable runtime. A general rule of thumb that is used in theory is a scenario set of 30. We adopt this rule and thus choose a scenario set of 30 per instance. Using the results of all 30 scenarios, we calculate the average performance and use that to evaluate all experiments.

To generate different scenarios, we need to define probability distributions for the disruptions. Unfortunately, there is limited data on the disruptions at HTM Aerotec. Therefore, we will estimate distributions and parameters based on the literature and estimations from production managers. For each scenario we will generate, based on these distributions, we will determine if that disruption occurs for each of these three. If none of the disruption types occur based on the distributions, we regenerate a scenario. That means that in one scenario, one or multiple disruptions can occur.

- Machine failure
  We assume that the machines have reached a constant failure state, where the failure rate does not depend on time. Based on this assumption, we can use the exponential distribution, which is a commonly used distribution for machine failure (Chakrabarty et al., 2016). To estimate the failure rate, we use historical data from the machines as we also used in Figure 1.1. From this, we can see that machines 538 and 539 have a total rate of machine non-availability due to machine failure of approximately 4.7% and 5.2%, respectively. Therefore, the failure rates for both machines are approximately 0,0006. We assume the machine failure's duration, or the repair time, takes a normal distribution with a mean of 85 hours and a standard deviation of 24 hours.
- Operator illness

Experiment	Options	Experiment settings
Best-Fit	1. Tardiness and moves	These experiments are run with a linear expansion method and a maximum computational runtime of 60 seconds.
	2. Similar timeslot	
	3. Ratio of hours left	
	4. Weekly ratio balance	
Expansion method	Linear	We run these experiments with the best-fit method as found in the previous experiment. Furthermore, we use a maximum computational runtime of 60 seconds.
	Exponential	
	Logarithmic	
	Percentage-based	
Runtime	10 seconds	In these experiments, we use the best-fit and expansion method as found in the previous experiments.
	30 seconds	
	60 seconds	
	120 seconds	
Sensitivity Analysis	**Options**	**Experiment settings and goal**
Due dates	Original due dates	This experiment investigates the impact of relaxed due dates on model performance.
	Moving original due dates to one week later	
	Moving original due dates to two weeks later	
Job mix	Range from 0.1 to 1 with increments of 0.1	In this experiment, we change labour times using a multiplication factor. This experiment aims to analyse the impact of different job mixes.

Table 5.1: An overview of all experiments

In Section 2.1, we note that downtime of machines due to operator illness requires the absence of multiple operators, which does not occur frequently. We assume a similar distribution and rate as for the machine failure, as this probability is not time-dependent. The duration of this disruption differs from the machine failure case. We assume a normal distribution with a mean of 48 hours and a standard deviation of five hours.

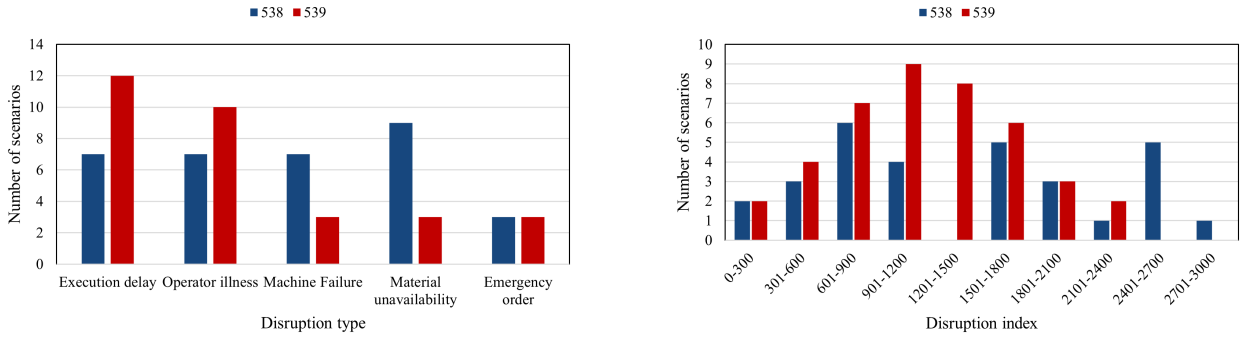
- Emergency order, execution delay and material unavailability
  For these three disruption reasons, the company planners assume a 20% possibility that one of these occurs each day. There, the probability for which of those three is equal for all three. Hence, we take a 0.2 probability that one of these occurs. If so, we randomly take one of these three. For the duration of the disruption, the emergency order has no duration. For the execution delay and the material unavailability, we assume a normal distribution with a mean of 10 hours and a standard deviation of 3 hours.

The moment at which the disruption occurs is generated randomly by choosing an index from the initial schedule. Using these probability distributions and indices, we generated thirty scenarios per instance. Figure 5.2 shows how the scenarios are distributed over the scenario types and the indices. The disruption index corresponds with the first operation in the initial schedule at which the disruption occurs. A complete overview of the generated scenarios can be found in Appendix A.

5.2 Best-Fit method

The best-fit method aims to determine the optimal position in the sequence that results in the best scheduling outcome. In this section, we evaluate four scoring methods and their respective formulas, as outlined in Section 4.4.4.

First, we analyse the value of the weight parameter ω on scheduling performance when using the tardiness-stability estimation method. The parameter ω balances our multi-objective: schedule stability and estimated tardiness. A value of 1 prioritises stability exclusively, while a value of 0 bases the score entirely



(a) Type

(b) Index

Figure 5.2: Overview of the number of scenarios per disruption type and index

on the estimated tardiness. We hypothesised that lower values of ω would result in lower tardiness at the expense of stability. However, Figure 5.3 shows minimal differences between the values of 0, 0.5 and 1. Moreover, these differences do not follow a consistent trend. For example, scenario 5 has a lower tardiness for weight 0, while scenario 28 has a lower tardiness for weight 1. This contradiction to the hypothesis can be explained by the limitations of our tardiness estimation. As described in Section 4.4.4, we estimate tardiness based on a simplified model which only evaluates 100 operations and considers shift hours as the only constraint. Furthermore, this model evaluates the tardiness based on inserting one job into the initial schedule. However, when inserting multiple jobs, these influence each other's tardiness. As the number of jobs to reschedule increases, the local estimation becomes less accurate as the influence of interdependence between jobs increases. This interaction and the limited constraints result in an estimation which can deviate heavily from the actual tardiness.

To analyse the influence of the interdependence of jobs and the quality of the estimation, we performed additional experiments. We constructed an instance in which the shift constraint was removed. This can significantly improve the tardiness estimate, as the interdependence between inserting operations does not result in potentially inserting operations during off-shift hours, causing downtime. With this experiment, we found a significant influence of the weight factor ω. An ω of 1, focusing on stability, provides us with an average tardiness of 4.05 hours and zero moved operations. However, an ω of 0, thus focusing on tardiness, provides us with an average tardiness of 0.75 hours and 25 moved operations. This is an improvement of 81.5% for tardiness. Therefore, we conclude that the accuracy of the tardiness estimation, and thus the effectiveness of this best-fit method, is heavily influenced by the presence of shift constraints. When shift constraints are removed or when shifts are significantly extended, the impact of interdependent operations on the estimation decreases, leading to more reliable tardiness estimates. In such contexts, our tardiness-stability estimation best fit method can be powerful for minimising tardiness while balancing stability. Moreover, this insight suggests that improvement of the estimation method, accounting for the interdependence and complex constraints, can broaden the applicability of this method. However, such improvements will likely increase computational runtimes.

Next, we compare the four methods of determining the score for the best fit. Figure 5.4 shows boxplots comparing the four methods in terms of schedule stability, tardiness, and utilisation. Figure 5.4(b) only presents data from machine 538, as machine 539 has zero tardiness across all methods. This is due to its more evenly distributed workload, as explained in Section 5.1. This workload distribution also explains the lower utilisation rate of machine 539. Additionally, the general trend of decreasing tardiness over the initial planning horizon, as mentioned in Section 5.1, is reflected in the utilisation pattern of machine 538. Specifically, Figure 5.5 shows a decrease of 10% average decline in utilisation across the planning horizon (scenarios are increasingly sorted based on their moment of breakdown in the initial planning) for all methods combined. This indicates that schedules generated earlier in the planning horizon, where due dates are tight, require higher utilisation to meet deadlines. Regarding schedule stability, the method

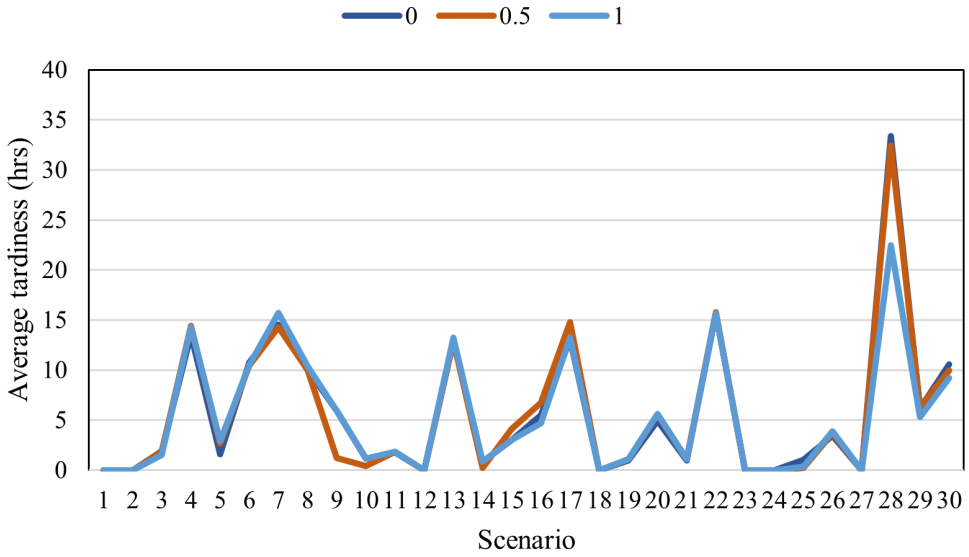
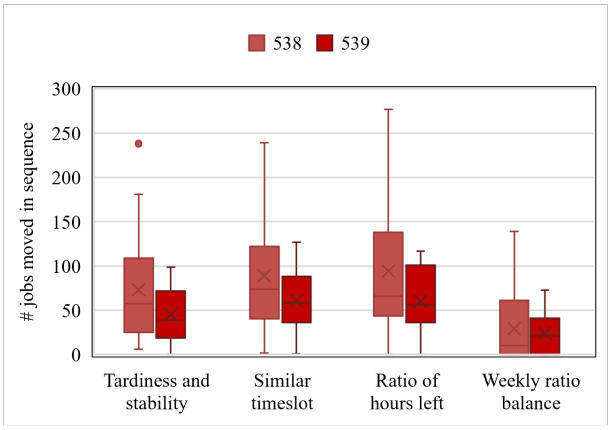


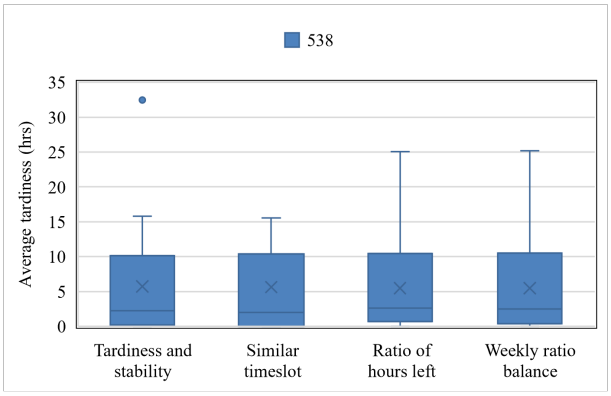
Figure 5.3: Tardiness of each scenario with different ω

using the weekly balance ratio, designed to balance manual and automatic labour, performs well. However, both the tardiness and utilisation of this method are lacking. In terms of tardiness, we see that the methods based on tardiness and stability and the similar timeslot perform well. The first of the two, based on tardiness and stability, slightly outperforms the second in terms of stability, but the utilisation rate of the similar timeslot method is better. The method based on the ratio of hours left has a poor stability score and tardiness.

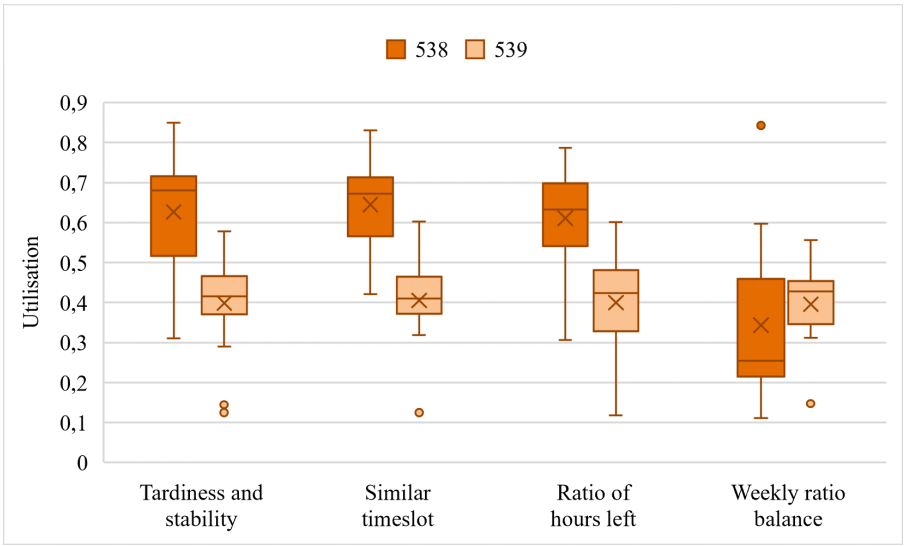
From these results, we conclude that the preferred scoring method heavily depends on the scheduling context. In situations where the quality of the tardiness estimation is low, inherently stable scoring methods are more reliable; therefore, the tardiness and stability estimation method should not be used. While other situations can benefit from this method. Additionally, in environments where utilisation is important due to bottlenecks, like at HTM Aerotec, the similar timeslot method proves best in balancing the tardiness with utilisation. Within the production process of HTM Aerotec, the 5-axis machines are the bottleneck. Therefore, maintaining a high utilisation rate and low tardiness on these machines is of high importance. Other machines have buffers. Therefore, in discussion with the production manager, we conclude that a method which outperforms others on tardiness and utilisation is preferred. Hence, we conclude that the method characterised by planning in a similar timeslot is the best solution. However, in possible future situations where these machines are not a bottleneck and other machines might not have the buffer to change planning easily, it would be worthwhile to consider the method based on the weekly ratio balance.



(a) Stability



(b) Tardiness



(c) Utilisation

Figure 5.4: Boxplots of the stability, tardiness, and utilisation of the schedule per method

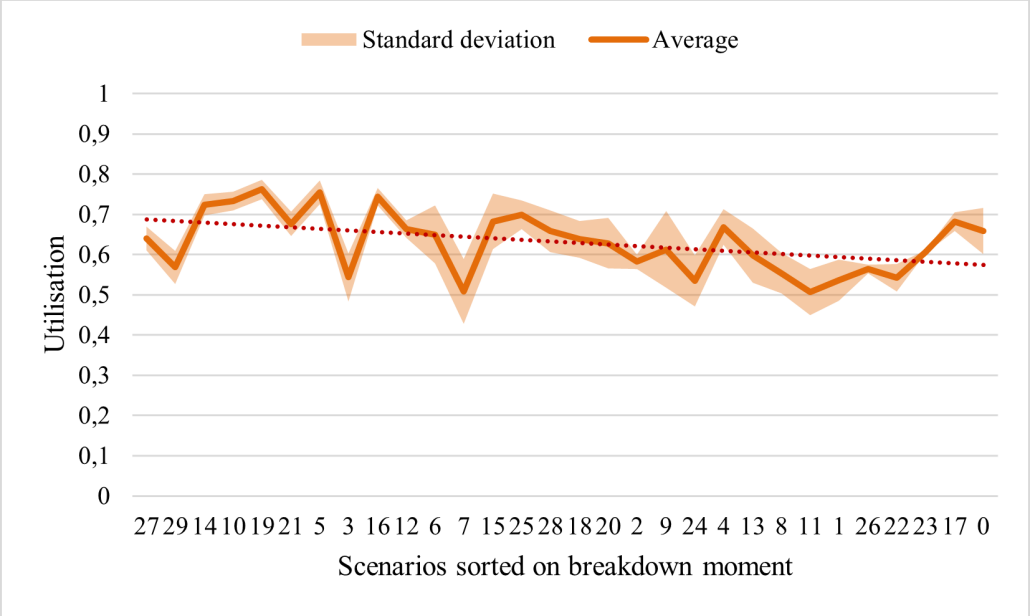


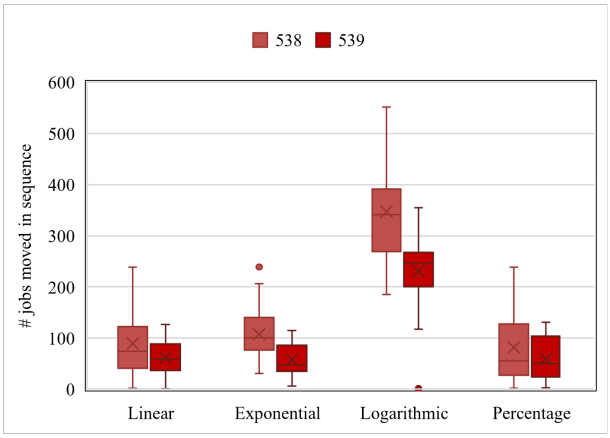
Figure 5.5: Average utilisation of the methods for machine 538 sorted on the initial breakdown moment

5.3 Time window expansion

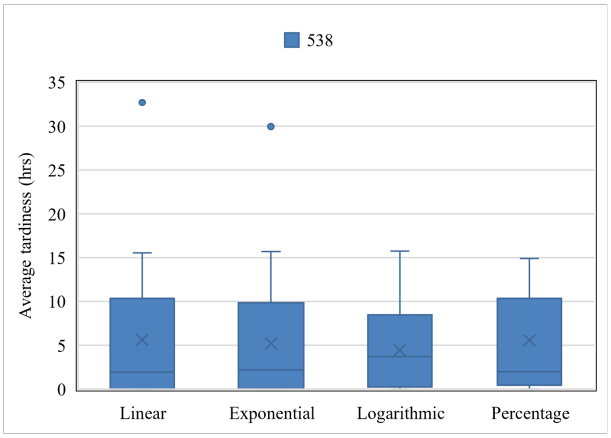
After each iteration, the time window is expanded. The time window determines the operations which are rescheduled, and thus, the expansion method influences how many jobs are rescheduled. In the previous section, we used the linear expansion method. In Section 3.3.2, we discussed four methods of time window expansion: linear, exponential, logarithmic and percentage-based. The formulas for expanding the upper bound can be found in Equations 4.30, 4.31, 4.32 and 4.33. As shown in Figure 4.4, different methods have larger expansions in early iterations (e.g. Logarithmic), while others have larger expansions in later iterations (e.g. percentage-based). We analyse these different methods to find which method best fits our scheduling problem. Figure 5.6 shows boxplots of the stability (a), tardiness (b) and utilisation (c) per expansion method. Again, the tardiness only includes data for machine 538, as the schedules for machine 539 do not result in any tardiness.

Overall, the differences between methods are minor. Notably, the logarithmic method decreases the stability without a significant improvement in tardiness. The percentage-based method has a favourable balance, resulting in a more stable schedule and tardiness. The utilisation, shown in Figure 5.6(c), shows no substantial differences, although we can see a slightly lower average for the linear method on machine 539 and a larger variability for the percentage-based method on machine 538. Again, we analysed these graphs with the production manager and operations manager. We concluded that the percentage-based method is most suited for HTM Aerotec in its current situation, due to the lowest maximum value. In cases where stability is important, this does not differ, as in terms of stability, this method performs similarly to the other methods. This conclusion shows the importance of operational practicality. Moreover, it underlines the dependence on the context.

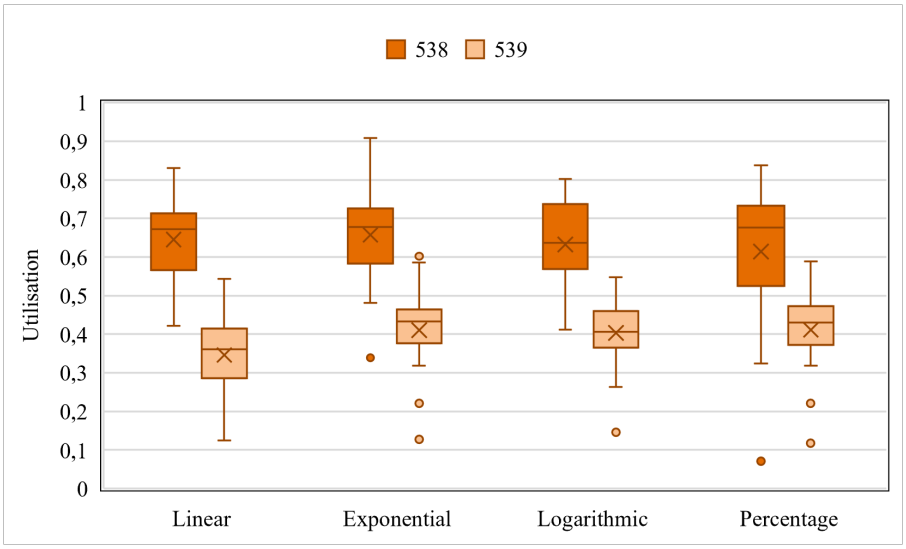
When examining the performance of the model when using the percentage-based method, we see an average utilisation of 63.5% and 41.1% for machines 538 and 539, respectively. On average, this leads to 106 and 69 production hours weekly. Especially, the production hours of machine 539 here are lacking, as in the past, the machines have produced 110 hours on average. However, as already discussed in Section 5.1 and Section 5.2, this can be due to the moment of disruption in some specific scenarios. Figure 5.7 shows the utilisation rates per scenario, sorted on the moment of disruption. In that graph, we can see that for machine 538, the utilisation rate is significantly higher when rescheduling early in the planning horizon. For machine 539, this is not the case. Again, this can be explained by the more evenly spread workload and absence of tardiness. HTM Aerotec will regenerate an initial schedule every two weeks, which therefore



(a) Stability



(b) Tardiness



(c) Utilisation

Figure 5.6: Boxplots of the stability, tardiness, and utilisation of the schedule per method

means that the disruption always occurs in those early stages of the schedule. Out of the thirty scenarios we sampled, this is the case for ten. The average utilisation of these on machine 538 is 74.9%, which is 126 weekly production hours. Hence, we can conclude that the performance of the model and the average production hours reached are higher than the average outcome of our experiments if used with a regularly updated initial schedule. Therefore, at HTM Aerotec with realistic scenarios, we can conclude that the model is capable of achieving acceptable production targets. On the other hand, if rescheduling is required near the end of the planning horizon, the model performs noticeably worse. This suggests that for late-disruption scenarios, alternative or more dynamic rescheduling models may be more appropriate.

In conclusion, while performance differences between expansion methods are relatively minor, as also shown in Section 5.2, this analysis reveals that the effectiveness of each method is sensitive to the timing of the disruption and the workload distribution. The percentage-based method is well-suited to HTM Aerotec's scheduling context, where disruptions occur early in the planning horizon. However, this approach may not generalise well without adaptation. To extend the applicability, future research should be done to see how these context dependencies can be eliminated.

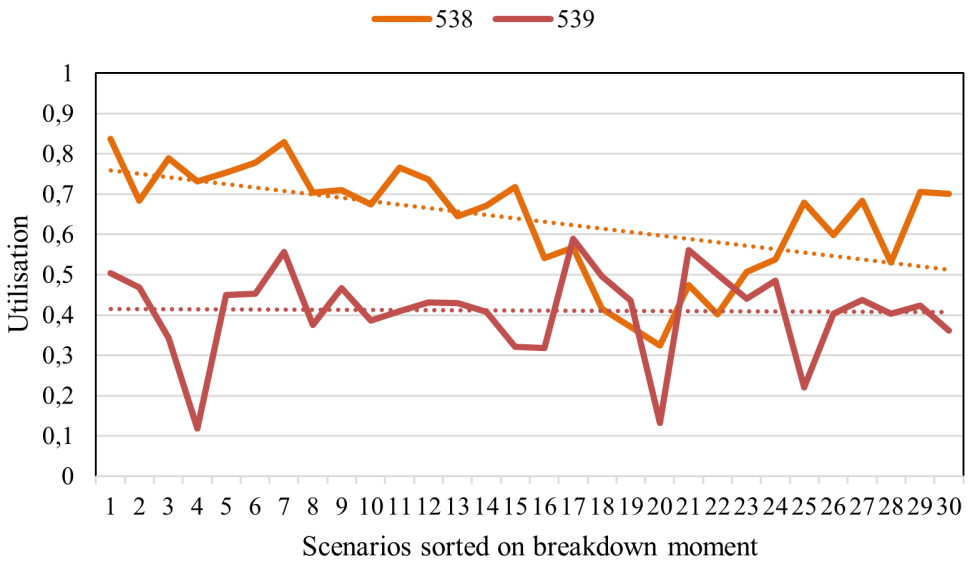
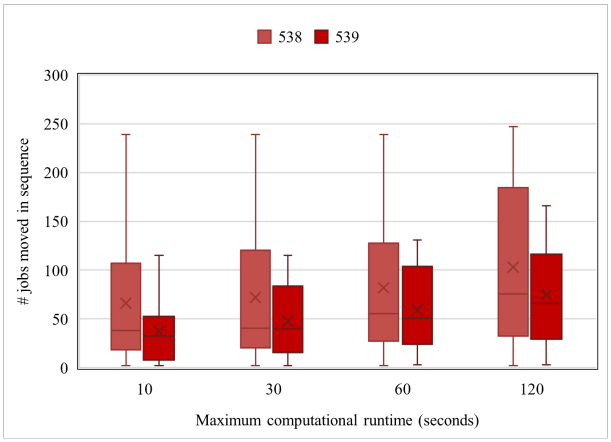


Figure 5.7: Utilisation of the percentage-based method sorted on the initial breakdown moment

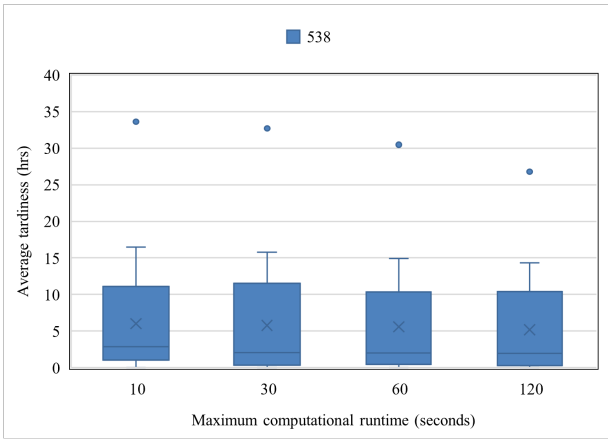
5.4 Maximum runtime

The maximum runtime determines the number of iterations the algorithm can run. Therefore, this limits the number of time window expansions and consequently the number of operations which are rescheduled. A longer runtime, therefore, might lead to worse stability. The decrease in stability with longer computational runtimes is primarily caused by the increased number of operations that are rescheduled with a more extended time window. As our best-fit method does not penalise sequence changes, operations are moved freely, thus increasing the instability. This is in contrast with our objective, as we want to minimise instability. Contrarily, the tardiness and utilisation may become better the more operations are rescheduled. Therefore, we need to find a balance between improved tardiness and decreased stability using the computational runtime. Figure 5.8 demonstrates this trend for all three metrics. We can see that our model has only slight differences in both tardiness and utilisation based on the runtime. When looking at the differences in performance between a runtime of 10 and 120 seconds for machine 538, we can see a clear imbalance. As the number of operations increased by around 60%, the tardiness decreased by only 17%. Between 60 and 120 seconds, these percentages are 30% more operations moved and 10% decreased tardiness. This imbalance shows that an increased runtime should only be considered if the cost of tardiness is significantly higher than the cost of schedule instability. When looking at the practical implementation of the model, discussions with the operator and production manager reveal that stability is valued because of the reduction in coordination. Therefore, for HTM Aerotec, we conclude that a maximum computational runtime of 120 seconds has too high a stability decrease.

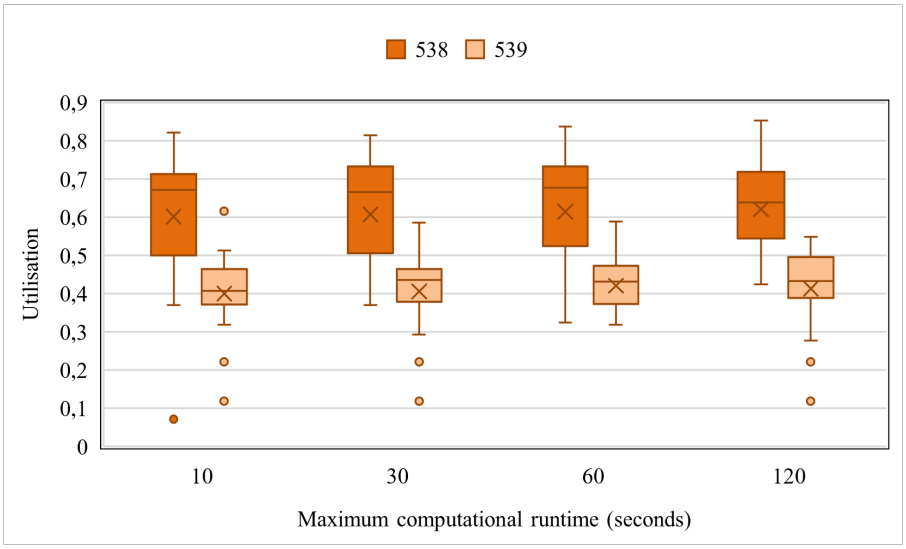
The differences between ten, thirty and sixty seconds are small. However, lower tardiness and higher utilisation are always preferred. Furthermore, from discussions with the operators, production manager and operations manager, we concluded that a runtime of sixty seconds would be sufficient. Additionally, the time an operator gains by running the algorithm ten seconds instead of sixty seconds is negligible compared to the work they do. Therefore, we conclude that a runtime of sixty seconds is the best setting for HTM Aerotec. As we performed the previous experiments with a maximum computational runtime of sixty seconds, this means that the average tardiness, utilisation and stability are identical to those discussed in Section 5.3. Future improvements of the model could include stability thresholds, limiting the runtime if this threshold is reached.



(a) Stability



(b) Tardiness



(c) Utilisation

Figure 5.8: Boxplots of the stability, tardiness, and utilisation of the schedule per maximum computational runtime

5.5 Comparison to the initial and manual schedule

To assess the effectiveness of the rescheduling approach, we compare its performance with both the initially generated and manually constructed schedules. This comparison enables us to evaluate whether the model resolves disruptions more effectively.

Utilisation data based on our input data for the manually constructed schedules is not available. However, we can compare scheduling logic through product timing. At HTM Aerotec, a rule of thumb is used: operations with a processing time of less than an hour are scheduled during shifts (day work), while operations exceeding two hours are allocated to night or weekend hours. Intermediate durations are assigned flexibly. Our model does not enforce these rules explicitly. A detailed analysis of the schedules reveals that at least 70% of the night and weekend work is done outside of shift hours, in some scenarios even 100%. This generally aligns with the manual planning. However, we can also see some products with very short processing times, like five minutes, being produced during those hours, while during the day some operations with longer processing times (e.g. 45 minutes) are planned. This can result in lower utilisation rates, as the machine accumulates less runtime overall, potentially leaving it without any scheduled operations before the next shift begins. Based on these findings, we conclude that our model does partially align with the manually made schedule, but it does have some differences. Furthermore, we can conclude that there

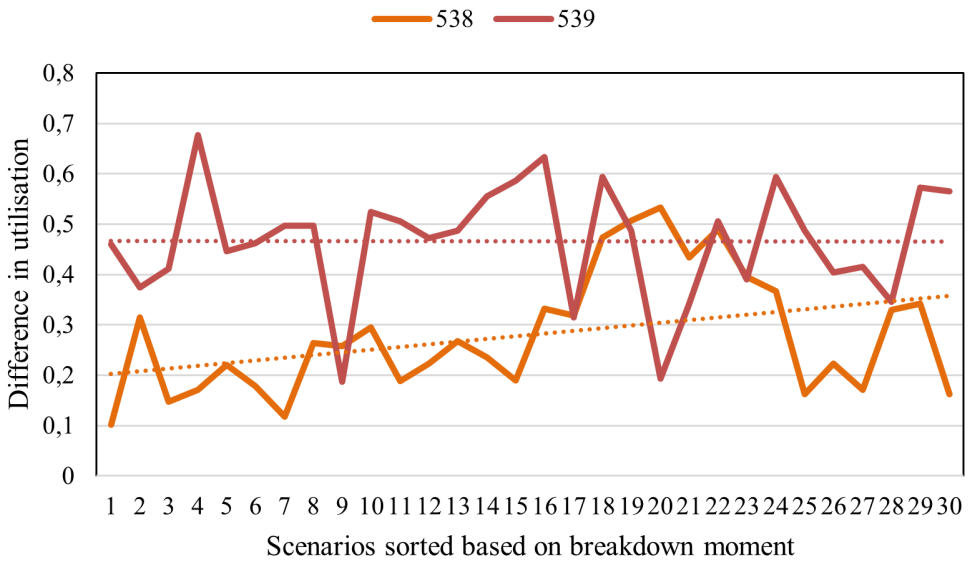
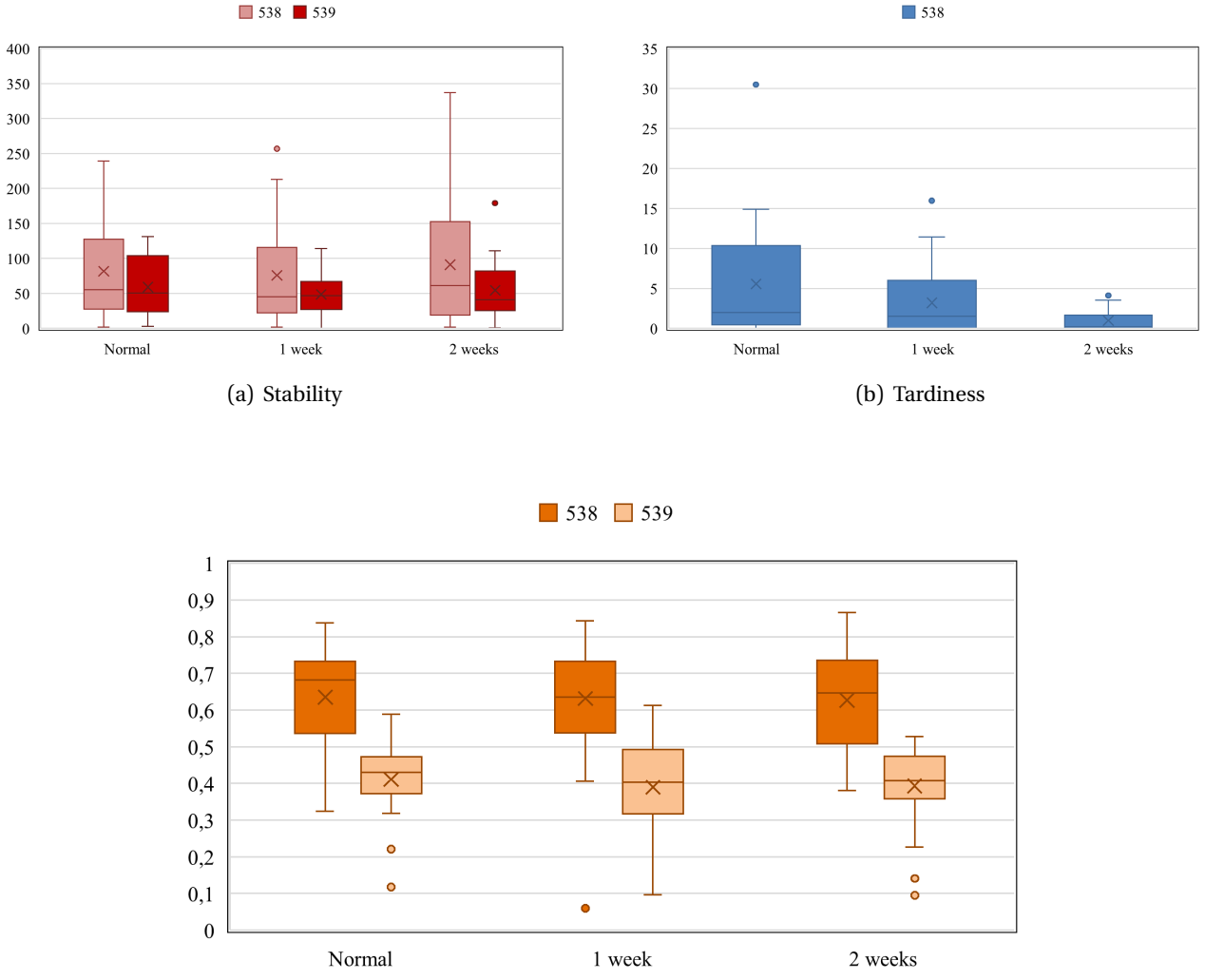


Figure 5.9: The difference between the initial schedule utilisation and repaired schedule utilisation for both machines

is room for improvement in our model, as swapping products with longer production times to produce outside shift hours would increase machine utilisation.

Next, we compare the utilisation of the initial schedule. For this, we calculate the utilisation of the corresponding week in that initial schedule. This serves as a baseline, as the machines should be able to reach this utilisation when no disruptions have occurred. Figure 5.9 then shows the difference in utilisation, which is the initial utilisation minus the utilisation of the rescheduled schedule. The vertical axis represents this difference, and the horizontal axis is sorted by the timing of the disruption. A smaller difference implies that the rescheduling model performs (more) similarly to the initial schedule. In the graph, we can see that our model maintains a high utilisation for machine 538, particularly when disruptions occur early in the planning horizon. This supports earlier conclusions that the performance is better when rescheduling at the start of the time horizon. In contrast, the performance of machine 539 consistently shows higher differences, therefore showing that our model has a large performance difference from the initial schedule. This occurs especially when the schedule is not as tight in terms of due dates. Furthermore, while the initial utilisation is above the goal of 83%, our proposed model averages 75% when rescheduled early in the planning horizon. From this, we can conclude that the use of the initial schedule is preferred over the use of our model.

Another point of interest is the difference in performance between scenarios that have comparable disruption moments. For example, we can see a 20% difference between the first and second scenario of machine 538. These differences are due to the timing of the disruption. Scenarios which have breakdowns at the start of the night or during the weekend have a lower utilisation because the model must wait until the start of a shift. We assumed disruption durations based on probability distributions which do not take into account working hours. For instance, if a breakdown is resolved at 11:00 on a Saturday, production only resumes on Sunday at 10:00, leading to an additional 23 hours of inactivity due to shift constraints. In reality, such downtime would have already been inherent in the duration of the disruption and therefore would not contribute to extra downtime in the manual planning.



(a) Stability

(b) Tardiness

(c) Utilisation

Figure 5.10: Boxplots of the stability, tardiness, and utilisation of the schedule with delays in the due dates

5.6 Sensitivity analysis

In this section, we aim to find out how the algorithm behaves under different circumstances than those used during the previous experiments. The biggest differences possible in our research context are changes in due dates and job mix. In Section 5.6.1 we analyse the influence of due dates, and in Section 5.6.2 we discuss the influence of the job mix.

5.6.1 Due dates

At the time we obtained the data to generate the initial schedule, HTM Aerotec was already behind schedule. As a result, the due dates in our problem instance are tight, with some items already past their due date. However, as illustrated in Figure 5.1, the high utilisation of the machines in the initial schedule suggests that HTM Aerotec is likely to get ahead of schedule within a few months. This effectively corresponds to a shift in due dates compared to the current situation. Since our objective function is partially based on tardiness, and thus indirectly influenced by due dates, it is important to analyse how changes in the due dates affect the performance of our approach.

As mentioned in Section 5.1, for the due date experiments, we use the same instances as before, but we modify the due dates of jobs. The due dates were shifted by one and two weeks later. This results in more flexibility and lower tardiness in the initial and repaired schedules. Figure 5.10 presents the boxplots for

the stability, tardiness and utilisation of the schedules. As expected, we can see a clear decrease in the average tardiness as due dates are postponed, with a 70% and 80% decrease when postponing one and two weeks, respectively. This is intuitive, as longer due dates allow more operations to be scheduled without violating the due date constraint. Strikingly, we see a modest decrease of 2% in utilisation for both machines. We can explain this by the added slack due to relaxed due dates. This allows the initial model to prioritise other objectives, and thus also changes the behaviour in rescheduling. However, this small reduction is not significant to the overall performance of the model. Additionally, the impact on the stability of our model does not show a clear trend. When looking specifically into the performance of single scenarios, we see that some have improved stability, while for others, this is significantly decreased. From this, we conclude that the stability of the rescheduling model is mostly governed by other factors, such as the disruption timing.

From these observations, we conclude that extended due dates do improve the tardiness, but do not significantly impact the model's performance in terms of stability or utilisation. As the tardiness improvement is inherent to relaxed due dates, we conclude that due dates do not have a significant impact on the performance of the model.

5.6.2 Job mix

The job mix refers to the variety of items to be produced and their associated labour and processing times. Different job mixes lead to varying utilisation rates for the machines in the initial schedule, as a higher proportion of manual labour reduces flexibility within the available shift hours. The current job mix required a set percentage of manual labour hours. Less flexibility occurs when these percentages increase, given that the total available manual labour hours is set. We analyse how such variations in the job mix affect the performance of our solution approach.

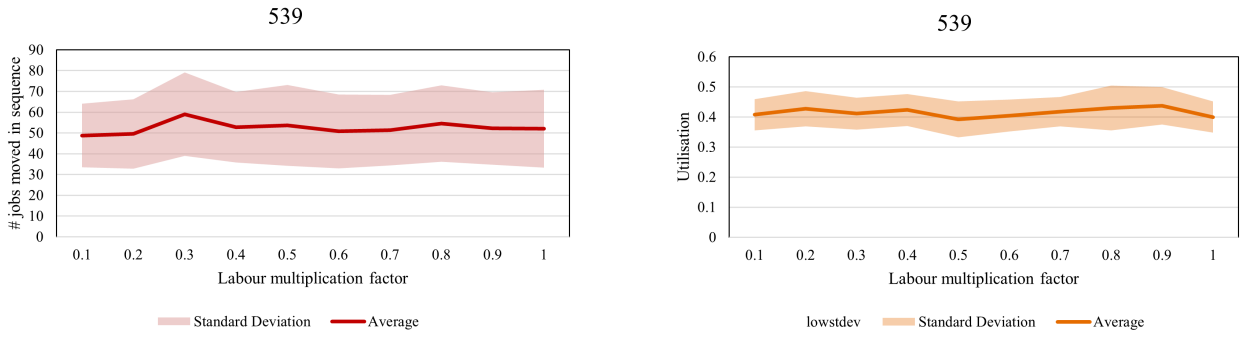
To analyse the behaviour of our algorithm with different job mixes, we perform experiments by varying the multiplication factor of the labour times from 0.1 to 1. This factor adjusts the weight of labour hours compared to automatic hours. We generate ten different initial schedules, each corresponding to a different factor, and run our model for each. In general, the average tardiness over these initial schedules decreases as the multiplication factor decreases. However, as the model made by Van Boxel (2024) has some randomness, this is not always the case. The results after rescheduling are therefore partially influenced by randomness in the initial schedules.

Figure 5.11 illustrates the stability and utilisation per labour multiplication factor of machine 539. In these graphs, we do not see any clear trends when changing the multiplication factor. This suggests that the labour multiplication factor does not have a significant impact on the performance of our model for this machine. Figure 5.12 shows the stability, tardiness and utilisation of machine 538 per multiplication factor. Again, tardiness and stability do not show a clear trend. However, the graph regarding utilisation does show a decline of 10% when increasing the labour factor. This is consistent with expectations. Increasing the labour time per operation shifts the balance between labour and automatic hours, making it more likely that the required number of manual tasks to have a similar utilisation cannot be performed, causing more downtime for the machines.

We conclude that the labour factor mainly impacts the utilisation. This impact highlights the dependency between shift hours and automation.

5.7 Generalisation to other machines

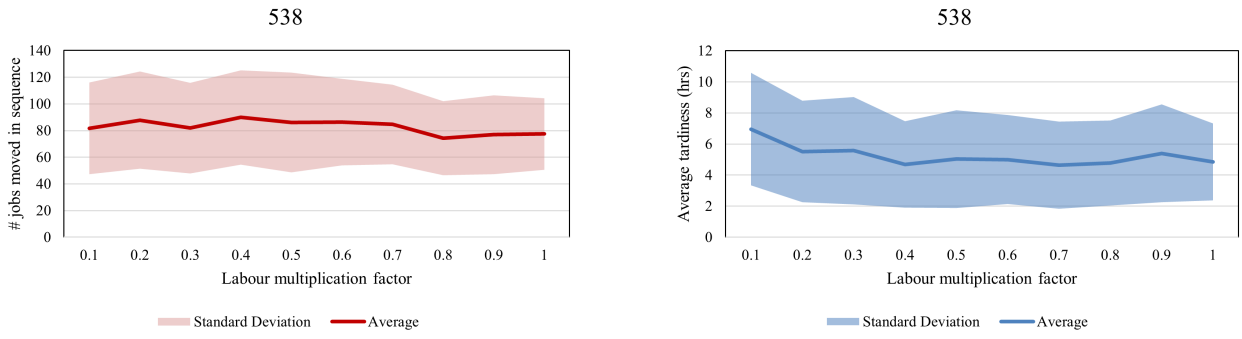
In our model, we incorporate machine-specific constraints such as dedicated fixtures, limited pallet storage capacity, and fixed pallet switching times. It is designed for an FMC with a pallet handling system. Consequently, the model can be applied to other machines that have similar characteristics. Figure 5.13 shows the performance of the model when used for machine 533. There, we can see that the utilisation for this machine lies above 70% on average, with good stability. This is a higher score compared to machines



(a) Stability

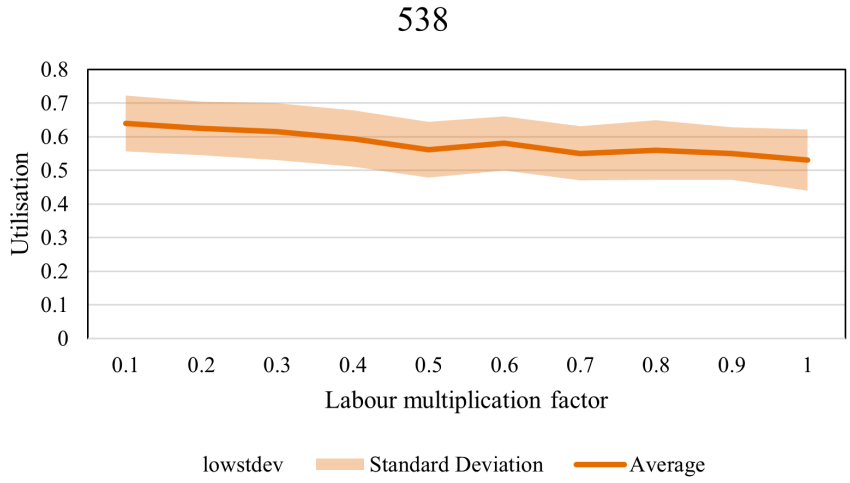
(b) Utilisation

Figure 5.11: Boxplots of the stability, tardiness, and utilisation of the schedule with different job mixes for machine 539



(a) Stability

(b) Tardiness



(c) Utilisation

Figure 5.12: Boxplots of the stability, tardiness, and utilisation of the schedule with different job mixes for machine 538

538 and 539. If we look at the tardiness (Figure 5.13(c)), we see a similar trend for machine 533 as for the other machines: after a certain period, the model is ahead of schedule and thus no tardiness occurs anymore. As demonstrated by its application to machine 533, the model is also adaptable to settings where the machining cell includes two machines, sharing a common pallet storage system. This adaptability can be extended to configurations involving additional machining cells with minimal adjustments.

Besides immediate practical generalisation, we can look at the logic and approach we used in this research for broader applicability. The percentage-based time window expansion is effective in settings where disruptions occur at the start of the planning horizon. This suggests that the model may perform well in contexts where rescheduling occurs frequently, requiring flexibility. However, the effectiveness of the scoring methods heavily depends on specific problem constraints. The scoring method based on tardiness and stability is expected to perform better in contexts without shift constraints, whereas the shift-time-based method performs well in contexts with strong job dependencies and shift constraints.

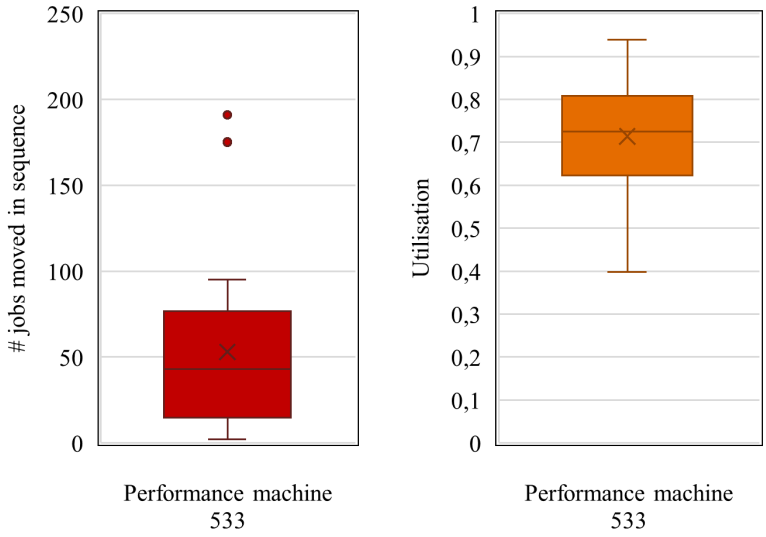
In summary, while the model was developed within a specific industrial setting, its main logic and approach provide a foundation that can be extended to a broader class of flexible machining environments. The successful generalisation will depend on the degree to which these environments share key characteristics with the HTM Aerotec context, particularly in terms of scheduling constraints. These results suggest that the logic of this model is not specific to the exact configuration of our machines, but depends on operational context.

5.8 Conclusion

In this chapter, we addressed the fifth research question by conducting a series of experiments to validate the proposed solution and assess model effectiveness. We developed problem instances for machines 538 and 539 that vary slightly in size, time horizon and job mix. These variations allowed us to evaluate model performance across diverse scenarios. Additionally, we introduced an instance for machine 533 to test the generalisability of the model beyond the initial experimental scope. The experiments focused on three points: the best-fit method, the expansion method and the maximum computational runtime. Furthermore, we conducted a sensitivity analysis to explore the influence of varying due dates and job mixes.

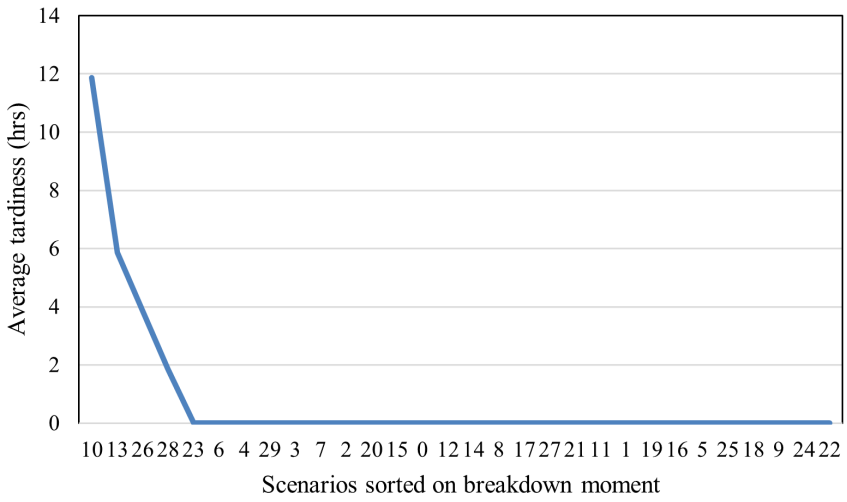
The weight parameter ω does not significantly impact the tardiness and stability-based scoring method. This is largely due to limitations in the estimation method. Among the four scoring methods, the similar timeslot method achieves the best balance between tardiness, utilisation and stability. Therefore, this is the preferred method. However, in contexts where stability is a higher priority, we recommend considering the weekly balance ratio method, as this method maintains better stability. Our findings indicate that the choice of time window expansion method has a limited impact on performance metrics. The percentage-based method offers the best balance between stability and tardiness, therefore making it the most suitable for HTM Aerotec. The effectiveness of the model is most present when disruptions occur early in the planning horizon. This aligns well with the scheduling frequency of HTM Aerotec. However, late disruptions show a reduction in the performance of the model, hence suggesting that other approaches are required in those cases. Using this method, we see a higher average utilisation rate of 74.9%, when updating the initial schedule regularly. Lastly, we analysed the maximum computational runtime of the model. We showed that longer runtimes slightly improve the tardiness and utilisation, but significantly reduce schedule stability. Differences between 10, 30 and 60 seconds are minor, but a runtime of 60 seconds yields better performance in terms of tardiness without overburdening the operators. Thus, a 60-second runtime offers an effective balance between performance and operational practicality. Future model improvements could include stability thresholds.

We conducted a sensitivity analysis to analyse the impact of due dates and the job mix on our model. Shifting the due dates results in a large decrease in tardiness, as expected. However, the stability and utilisation do not exhibit a clear trend. We conclude that the stability and utilisation of the schedule are mostly governed by other factors, and thus, the due dates do not have a significant impact. From the analysis of the



(a) Stability

(b) Utilisation



(c) Tardiness

Figure 5.13: Boxplots of the stability, tardiness, and utilisation of the schedule with delays in the due dates

job mix, we conclude that this mainly influences the utilisation of the machines. The stability and tardiness are not influenced heavily. This highlights the dependency on shift hours and automation.

Additionally, we compared the model's performance with both the initial and manually constructed schedules. When comparing the model to the manually constructed schedules, we can conclude that at least 70% of the night and weekend operations are planned during those moments. While there is room for improvement, this does show that our model mostly adheres to the rules of thumb used in manual planning. Moreover, we conclude that our model consistently has lower utilisation rates than the initial schedule. Hence, using the initial schedule when possible is preferred. However, we also conclude that our model includes downtime, which in practice would be inherent to the planning, thus giving a bias to our results.

Lastly, we analysed the generalizability of the model. Comparison with the results of machine 533 reveals that the model is adaptable to other FMC configurations, with multiple machines in one cell. Furthermore, we conclude that the percentage-based time window expansion method is effective if disruption often occurs at the start of a planning horizon. However, the characteristics of the scheduling context heavily influence the performance. Moreover, when shift constraints are absent, a scoring method based on tardiness and stability proves more effective than time-based approaches.

6| Implementation

In this chapter, we outline the implementation requirements. We do this by answering the sixth sub-question from Section 1.3:

> *What technical, organisational, and procedural requirements are necessary to successfully implement the scheduling algorithm in a production environment?*

In Section 6.1 we discuss the data required as input for the model. Section 6.2 explains how the model should be used.

6.1 Data availability

To ensure that the model can function properly and output a feasible schedule, the input information should be correct. As mentioned in Section 2.3, the company does not always keep track of all of the required input information or does not store this in uniform places for all orders. Van Boxel (2024) described methods that would greatly reduce the time needed to look up this input. During the implementation phase of this research, we added information to the ERP system to ensure that this input (e.g. dedicated fixtures and multi-fixturing) is available at the correct places. Additionally, to complete the required input, an overview was made of all fixtures required for all items in production. This overview stores per item what fixtures are required for production per machine and how many of these fixtures are available. Especially in the case of dedicated fixtures, this is important, as these are a constraint on how many of those items can be planned in parallel on the machines. The company should keep this overview up-to-date if new dedicated fixtures are made for new items.

The model is adapted to ensure it can run either using information directly from the ERP system, as well as a schedule list as proposed by Van Boxel (2024). The information from Glovia is obtained using an API connection. This produces multiple data frames, which are then transformed and filtered to obtain the needed information. This transformation results in data frames which are of the same format as the schedule list. Therefore, only loading the data differs, making the code used for data transformation easy to understand, such that the company can update this easily when needed.

6.2 Using the model

In Section 4.5, we discussed the entire rescheduling process. In this process, we require the planner to use the solution of Van Boxel (2024), and the operator to use our proposed solution. Therefore, in this section, we first explain how the model of Van Boxel (2024) can be used and how the adapted output of the model should be interpreted. Then, we explain how our rescheduling approach can be used by the operator.

Over the past months, together with the Fraunhofer Innovation Platform, HTM Aerotec has been working on designing a prototype used at the machines to keep track of what items have been produced and the quantity. The prototype includes an interface that accesses the schedule list, and where operators can keep track of the quantities produced per item. The main page of the interface can be found in Appendix B.

As mentioned in Section 6.1, we adapted the model of Van Boxel (2024) to enable it to build the initial schedule directly from data in the ERP system. For this, the script HomeFile.py needs to be run. Once the model is finished running, the schedule is automatically written to a cloud from where the prototype can extract it. The prototype will be adapted such that from the interface, the schedule can be loaded, showing operators exactly which items should be produced and when. The output has a similar format to Table 6.1. The file shows in the first three columns the item and operation number, and description, respectively. These columns provide information for the operator. The fourth column shows what part of

Item	Operation	Description	Type	Start time	End time
1	1	product A	Setup	7:00	8:10
1	1	product A	Labour	8:10	8:14
1	1	product A	Run	8:14	8:47
2	1	product B	Labour	8:14	8:35
2	1	product B	Run	8:47	10:31
1	2	product A	Labour	8:35	8:59

Table 6.1: An example of the output from the algorithm

that operation the operator needs to perform. This can either be setup, labour or run. Setup means that of that item and operation, no other product has been made yet. Therefore, the operator needs to put the correct settings in the machine and has to prepare the machine. Labour is about ensuring that products are fixed on pallets and then put into the machine. For the lines that are producing, the operator does not have to do anything. There, the machine will automatically run. The operator can follow the file from top to bottom to find what steps should be taken at what moment in time.

To run the reactive model we designed, the operator can click on the button 'Regenerate schedule'. The interface will open a new page. There, the reason for rescheduling can be selected and the expected duration can be included. These are the input values as explained in Section 4.1. Then, the operator can run the model. The resulting schedule has the same format as shown in Table 6.1. This schedule automatically replaces the schedule initially used in the prototype. Therefore, the interface can be used in the same manner as with the initial schedule. This page of the interface still has to be made. We discussed the requirements and input fields, and the button has already been added as can be seen in Figure 1.

6.2.1 User acceptance

Due to limitations in the approval of our ethics review, we have not been able to conduct formal user acceptance tests (e.g., interviews, questionnaires). However, we gathered feedback through conversations with the production manager, operator and planner. These discussions reflected a general positive view on the proposed solution. During these conversations, the involved users recognised the added value and expressed interest in the implementation of the solution. Moreover, we have discussed the implementation plan and steps required to ensure alignment with other processes within the company. Initial steps of this plan have already been taken, including making a list of required and wanted changes in the prototype.

While this form of validation for user acceptance is informal, it provides a strong indication that the tool aligns with the needs of the company. The support of management and the first steps taken towards implementation increase the likelihood of adoption. This initial acceptance forms a solid foundation for further development and integration. Additionally, this early engagement from key users is essential for long-term adoption and process integration.

7| Conclusions & Recommendations

In this last chapter, we aim to answer the final sub-question and the main research question of this research. The final subquestion is:

> *What are the key findings, practical recommendations, and future research opportunities derived from this study?*

Section 7.1 discusses the conclusions and answers the main research question. Then, in Section 7.2 we discuss recommendations for the company based on this research. Next, Section 7.3 discusses the limitations of this research and provides topics for future research. Lastly, in Section 7.4 we discuss the scientific and practical contribution of this research.

7.1 Conclusions

Disruptions such as machine breakdowns can severely impact production efficiency. This research addressed the need for reactive scheduling strategies to mitigate such disruptions in a practical and time-efficient manner. We aimed to solve the core problem, which was the long runtime of the current scheduling algorithm. Running this algorithm after disruptions would take too long. To address this challenge, we formulated the following research question:

> *How can a scheduling algorithm be effectively designed to mitigate scheduling infeasibilities after disruptions and optimise production time in production schedules for 5-axis machines within a reasonable time?*

This algorithm reschedules operations within a time window, based on a best-fit position within a sequence. Identifying which position is the best fit is done by finding positions that were originally scheduled during a similar period. To effectively decide the length of the time horizon, and thus the number of jobs we reschedule, we implemented the use of expanding time windows using a percentage-based method. The model expands the time window for each iteration until the computational runtime limit of sixty seconds is reached. Using these settings, the model has the following performance:

- **Stability**: The model limits changes in the schedule, reaching an average stability of 82 and 59 jobs moved for machines 538 and 539, respectively.
- **Tardiness**: The model reaches an average tardiness of 5.5 hours per operation at machine 538, and machine 539 has zero tardiness.
- **Utilisation**: The average utilisation rate is 61.1% and 41.1%, leading to 103 and 70 production hours for machines 538 and 539, respectively. However, scenarios where disruption occurred in the early stages of the planning horizon reach an average utilisation of 75%, leading to 126 hours.

Based on the utilisation metric, we concluded that using the initial model is preferred. However, our rescheduling approach does perform sufficiently well in cases where the initial model is infeasible due to disruptions. Additionally, we concluded that the best fit and expanding time window methods should be context dependent; planning constraints and the moment of disruption influence the performance of these metrics. In the current situation at HTM Aerotec, the settings using the similar time slot and percentage-based methods are preferred. The performance is moderately affected by differences in due dates and the job mix; only the tardiness is decreased when due dates are relaxed. The other metrics do not signify a clear trend and thus do not impact the model performance. We did conclude that there is room for improvement, as the schedules do not always have the most efficient division in operations done during and outside shift hours.

To evaluate the generalisability of the model, we tested it on another five-axis machine with a different configuration. The model gave similar performance levels, thus suggesting that the approach is generalisable to other FMCs with different configurations. Additionally, the percentage-based time window expansion method is effective when early planning horizon disruptions occur frequently, while scoring methods to determine the best fit are shift-dependent.

To ensure the effectiveness of the proposed model, the required input data should be available. For this, we updated the ERP system and built a connection to this system with our model. Additionally, we propose a new rescheduling process. In this process, every two weeks, a new initial schedule is generated. If a disruption occurs, causing the initial schedule to become infeasible, then our rescheduling process is a guide to obtaining an effective schedule within a reasonable time. Furthermore, we outlined how the model can be used, such that the implementation of the model is eased. These implementation efforts include the addition of fixture data to the ERP system, the development of an API-based data connection, and integration with a prototype interface used by operators. By aligning model inputs with ERP data structures and enabling real-time data flow, the model becomes usable in daily operations with minimal manual effort.

In conclusion, the proposed model effectively addresses the main research question by enabling fast rescheduling. The model balances computational efficiency, performance and practical feasibility, therefore offering a practical solution for HTM Aerotec's production process.

7.2 Recommendations

Based on the conclusions drawn, we make the following recommendations to HTM Aerotec to maximise the practical impact of this research:

- Implement the proposed scheduling process
  Implementing the scheduling process, using the model of Van Boxel (2024) and our proposed model, will reduce the workload for the planning department and lead to more efficient utilisation of machine capacity. This will reduce the bottleneck at the five-axis machines.
- Evaluate the model with operators, planners and the production manager
  Implementing the model will change the daily workflow for operators, planners and the production manager, and may reveal unforeseen practical challenges. Therefore, we recommend involving these employees early on in the implementation process. Regular evaluation and feedback moments can help identify necessary adjustments.
- Evaluate the impact of implementing the proposed model
  The proposed model influences the scheduling of other processes and machines. The improvements on the five-axis machines may result in a reallocation of scheduling pressure to subsequent operations or machines. In this research, this impact is not evaluated or taken into account. Without this perspective, improvements in the scheduling of the 5-axis machines may shift the bottleneck to other operations. Therefore, an evaluation of this impact is required to prevent a potential ripple effect.
- Provide clear instructions for the use of the prototype interface
  The prototype interface enables operators to interact with the generated schedules and initiate rescheduling in case of disruptions. To ensure successful adoption, we recommend providing clear instructions for operators.

With these recommendations, we aim to ensure the successful implementation and long-term adoption of the model within HTM Aerotec. Additionally, these recommendations aim to increase alignment between planning assumptions and operational reality. By taking these steps, HTM Aerotec can maximise the impact of this research on its production efficiency.

7.3 Limitations and future research

In Chapter 4, we present a model based on literature, and Chapter 5 shows the required parameters and settings and validates the performance of the model. However, our research does have limitations and presents possibilities for further research. To improve practical efficiency, we address these below.

- Evaluate other repair algorithms
  In this research, we chose a basic insertion-based algorithm. This choice was based on literature (Section 3.3.2), runtime constraints and because of its simplicity in implementation. Other algorithms, like GA, could result in a better schedule, but this would be at the cost of computational runtime and is more complex. This would increase the difficulty for HTM Aerotec to maintain the algorithm and implement it. However, this trade-off should be evaluated against the improved efficiency to be able to conclude if our choice for a faster and easier solution outweighs the loss of efficiency.

- Extend the scope to the internal supply chain
  Currently, the solution approach focuses on a single machine. This optimises planning for one machine, but could cause the shift of the current bottleneck to other parts of the production process. To address this, the scope of the research should be extended to include the entire internal supply chain. This would turn the scheduling problem into a job-shop scheduling problem, as explained in Section 3.1, and would allow for better coordination and alignment of the complete production flow.

- Validate model performance with real-world data
  The experimental analysis in this research relies on scenario-based simulations. These scenarios are generated using estimated stochastic parameters, which may not fully reflect the real production process. Therefore, further analysis of the actual production data is required to validate the model's performance and its practical effectiveness.

- Tool constraints
  During this research, we investigated the possibility of including tool constraints. This requires access to the database where all information on tools is stored. This is a possibility; however, establishing this connection within the time frame of this research was not possible. Therefore, this model does not completely reflect all constraints as they occur in reality. Adding this connection would improve our model's validity.

- Solution choices
  In discussion with the operators and production manager, we concluded that they have preferences in the production sequence. These preferences vary and depend on the specific items. Therefore, it might be beneficial to provide the user of the proposed planning tool with multiple schedules. From these schedules, the user can then choose the preferred schedule. In our model, only one schedule is outputted. Therefore, further research could look into how different schedules should be generated.

Addressing these limitations in future research will help to improve the robustness, realism, and effectiveness of the scheduling model. In practice, extending the scope of the planning process could be a substantial benefit for HTM Aerotec. Furthermore, in the scientific field, model validation with actual data will expand the base of our model.

7.4 Scientific and practical contribution

In this section, we highlight the scientific and practical contributions of this research. This research offers both a scientific contribution to the scheduling literature and practical value for industrial implementation, particularly in the context of flexible manufacturing cells (FMCs). To our knowledge, this is the first study to design and validate a reactive rescheduling model specifically for FMCs, integrating expanding time windows with strict computational runtime constraints. Moreover, we have proposed a percentage-based time window expansion method. This method expands time windows with a set percentage based

on the current time window size, therefore distinguishing itself with easier comprehension and implementation. Moreover, we designed four scoring methods to calculate the best insertion position. These methods can be used in insertion-based heuristics, where the Best-Fit method is used. Lastly, we evaluated the model's generalisability by applying it to other FMCs with varying configurations within HTM Aerotec, showing the broader applicability of the proposed model, using the percentage-based time window expansion and time-slot-based best-fit approaches.

The practical contribution consists of three parts. First, we identified and implemented necessary adaptations to enable the integration of the model by Van Boxel (2024) into the production environment at HTM Aerotec. These include modifications in the ERP system, an updated fixture inventory, and a direct connection between the model of Van Boxel (2024) and the ERP system. This integration, along with the implementation of our rescheduling model, reduces the manual planning effort. Furthermore, the integration with a prototype interface allows operators to initiate rescheduling independently, enhancing the flexibility and responsiveness of planning processes. The implementation of the proposed scheduling process also establishes a structured process and helps formalise and standardise scheduling procedures.

REFERENCES

Adil, H., & Shaw, D. (2024). Effective Planning and Scheduling in Volatile, Very High Mix and Low Volume Discrete Manufacturing Environment. *International Journal of Modern Engineering Research (IJMER)*, *14*(6). https://www.researchgate.net/publication/385818272_Effective_Planning_and_Scheduling_in_Volatile_Very_High_Mix_and_Low_Volume_Discrete_Manufacturing_Environment

Al-Hinai, N., & Elmekkawy, T. Y. (2011). Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm. *International Journal of Production Economics*, *132*(2), 279–291. https://doi.org/10.1016/J.IJPE.2011.04.020

Baykasoğlu, A., & Karaslan, F. S. (2017). Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach. *International Journal of Production Research*, *55*(11), 3308–3325. https://doi.org/10.1080/00207543.2017.1306134

Baykasoğlu, A., Madenoğlu, F. S., & Hamzadayı, A. (2020). Greedy randomized adaptive search for dynamic flexible job-shop scheduling. *Journal of Manufacturing Systems*, *56*, 425–451. https://doi.org/10.1016/J.JMSY.2020.06.005

Blackburn, J. D., Kropp, D. H., & Millen, R. A. (1985). MRP system nervousness: Causes and cures. *Engineering Costs and Production Economics*, *9*(1-3), 141–146. https://doi.org/10.1016/0167-188X(85)90021-7

Chakrabarty, A., Mannan, S., & Cagin, T. (2016). Equipment Failure. *Multiscale Modeling for Process Safety Applications*, 309–338. https://doi.org/10.1016/B978-0-12-396975-0.00007-3

Cheng, Z., Wang, L., Tang, B., & Li, H. (2022). Research on Rescheduling of Microwave Components Flexible Job Shop with Disturbance. *Lecture Notes in Electrical Engineering*, *803*, 538–546. https://doi.org/10.1007/978-981-16-6328-4{_}56

Cowling, P., & Johansson, M. (2002). Using real time information for effective dynamic scheduling. *European Journal of Operational Research*, *139*(2), 230–244. https://doi.org/10.1016/S0377-2217(01)00355-1

Dawande, M., Geismar, H. N., Sethi, S. P., & Sriskandarajah, C. (2005). Sequencing and scheduling in robotic cells: Recent developments. *Journal of Scheduling*, *8*(5), 387–426. https://doi.org/10.1007/S10951-005-2861-9/METRICS

Elmaraghy, H. A., Abdallah, I. B., & Elmaraghy, W. H. (1998). On-Line Simulation and Control in Manufacturing Systems. *CIRP Annals*, *47*(1), 401–404. https://doi.org/10.1016/S0007-8506(07)62861-3

Ghaleb, M., & Taghipour, S. (2023). Dynamic shop-floor scheduling using real-time information: A case study from the thermoplastic industry. *Computers & Operations Research*, *152*, 106134. https://doi.org/10.1016/J.COR.2022.106134

Hamasha, M. M., Alazzam, A., Hamasha, S., Aqlan, F., Almeanazel, O., & Khasawneh, M. T. (2015). Multimachine flexible manufacturing cell analysis using a Markov chain-based approach. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, *5*(3), 439–446. https://doi.org/10.1109/TCPMT.2015.2394232

Hamzadayi, A., & Yildiz, G. (2016). Event driven strategy based complete rescheduling approaches for dynamic m identical parallel machines scheduling problem with a common server. *Computers & Industrial Engineering*, *91*, 66–84. https://doi.org/10.1016/J.CIE.2015.11.005

Heerkens, H., & Van Winden, A. (2017). *Solving Managerial Problems Systematically* (J.-W. Tjooitink, Trans.; 1st edition). Noordhoff.

Henning, G. P., & Cerdá, J. (2000). Knowledge-based predictive and reactive scheduling in industrial environments. *Computers & Chemical Engineering*, *24*(9-10), 2315–2338. https://doi.org/10.1016/S0098-1354(00)00589-5

Holguin Jimenez, S., Trabelsi, W., & Sauvey, C. (2024). Multi-Objective Production Rescheduling: A Systematic Literature Review. *Mathematics 2024, Vol. 12, Page 3176*, *12*(20), 3176. https://doi.org/10.3390/MATH12203176

Kuo, Y., & Li, D. X. (2024). Identical Parallel Machine Scheduling Problem with Additional Resources and Partial Confirmed Orders in Make-to-Stock Strategy. *Applied Sciences 2024, Vol. 14, Page 6736*, *14*(15), 6736. https://doi.org/10.3390/APP14156736

Kuster, J., Jannach, D., & Friedrich, G. (2010). Applying Local Rescheduling in response to schedule disruptions. *Annals of Operations Research*, *180*(1), 265–282. https://doi.org/10.1007/s10479-008-0488-x

Li, G., Xing, L., & Chen, Y. (2017). A hybrid online scheduling mechanism with revision and progressive techniques for autonomous Earth observation satellite. *Acta Astronautica*, *140*, 308–321. https://doi.org/10.1016/j.actaastro.2017.08.011

Li, Y., Tao, Z., Wang, L., Du, B., Guo, J., & Pang, S. (2023). Digital twin-based job shop anomaly detection and dynamic scheduling. *Robotics and Computer-Integrated Manufacturing*, *79*, 102443. https://doi.org/10.1016/J.RCIM.2022.102443

Liang, C., Zhang, Y., Hu, X., & Gen, M. (2019). Based on hybrid driven rolling window strategy for quay cranes and container trucks scheduling. *Proceedings of International Conference on Computers and Industrial Engineering, CIE*, *2019*, 2019.

Liu, Z., Jiang, Z., & Wu, L. (2023). Rescheduling Approaches for Ship Outfitting Pallet Sorting Operations. *ACM International Conference Proceeding Series*, 141–146. https://doi.org/10.1145/3613347.3613369

Lou, P., Liu, Q., Zhou, Z., Wang, H., & Sun, S. X. (2012). Multi-agent-based proactive-reactive scheduling for a job shop. *International Journal of Advanced Manufacturing Technology*, *59*(1-4), 311–324. https://doi.org/10.1007/S00170-011-3482-4/METRICS

Luo, Y., Zhang, J., & Li, G. (2022). Design and Research of Flexible Machining Cell(FMC)-Take Machining the Connecting Rod Parts as an Example. *Proceedings - 2022 International Conference on Cloud Computing, Big Data and Internet of Things, 3CBIT 2022*, 311–315. https://doi.org/10.1109/3CBIT57391.2022.00071

Narayanan, A., & Robinson, P. (2010). Evaluation of joint replenishment lot-sizing procedures in rolling horizon planning systems. *International Journal of Production Economics*, *127*(1), 85–94. https://doi.org/10.1016/J.IJPE.2010.04.038

Palombarini, J., & Martínez, E. (2012). SmartGantt – An interactive system for generating and updating rescheduling knowledge using relational abstractions. *Computers & Chemical Engineering*, *47*, 202–216. https://doi.org/10.1016/J.COMPCHEMENG.2012.06.021

Palombarini, J. A., & Martinez, E. C. (2019). Closed-loop Rescheduling using Deep Reinforcement Learning. *IFAC-PapersOnLine*, *52*(1), 231–236. https://doi.org/10.1016/J.IFACOL.2019.06.067

Pfund, M. E., & Fowler, J. W. (2017). Extending the boundaries between scheduling and dispatching: hedging and rescheduling techniques. *International Journal of Production Research*, *55*(11), 3294–3307. https://doi.org/10.1080/00207543.2017.1306133

Pujawan, I. N. (2004). Schedule nervousness in a manufacturing system: A case study. *Production Planning and Control*, *15*(5), 515–524. https://doi.org/10.1080/09537280410001726320;SUBPAGE:STRING:FULL

Qiu, J., Liu, J., Peng, C., & Chen, Q. (2024). A novel predictive–reactive scheduling method for parallel batch processor lot-sizing and scheduling with sequence-dependent setup time. *Computers & Industrial Engineering*, *189*, 109985. https://doi.org/10.1016/J.CIE.2024.109985

Sethi, S. P., Sriskandarajah, C., Sorger, G., Blazewicz, J., & Kubiak, W. (1992). Sequencing of parts and robot moves in a robotic cell. *International Journal of Flexible Manufacturing Systems*, *4*(3-4), 331–358. https://doi.org/10.1007/BF01324886/METRICS

Shariatmadari, M., & Nahavandi, N. (2020). A new resource buffer insertion approach for proactive resource investment problem. *Computers & Industrial Engineering*, *146*, 106582. https://doi.org/10.1016/J.CIE.2020.106582

Shin, J. H., Yu, J. M., Doh, H. H., Kim, H. W., & Lee, D. H. (2020). Batching and scheduling for a single-machine flexible machining cell with multi-fixturing pallets and controllable processing times. *International Journal of Production Research*, *58*(3), 863–877. https://doi.org/10.1080/00207543.2019.1602742

Sligting, N. (2024). Designing a complex order acceptance tool at Limis B.V.

Slomp, J., & Gupta, J. N. (2024). Lean Planning & Control in a High-Variety/Low-Volume Environment. *IFIP Advances in Information and Communication Technology*, *681*, 109–117. https://doi.org/10.1007/978-3-031-63265-5{_}9/FIGURES/4

Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Predictive-reactive strategy for identical parallel machine rescheduling. *Computers & Operations Research*, *134*, 105372. https://doi.org/10.1016/J.COR.2021.105372

Usman Nisar, M., Ma'ruf, A., Cakravastia, A., Halim, A. H., & Program, G. (2024). A Model of Proactive-Reactive Job Shop Scheduling to Tackle Uncertain Events with Greedy Randomized Adaptive Search Procedure. *Journal of Robotics and Control (JRC)*, *5*(6), 1626–1651. https://doi.org/10.18196/JRC.V5I6.22208

Van Boxel, N. (2024). An optimisation approach to schedule CNC machines with pallet handling system.

Vieira, G. E., Herrmann, J. W., & Lin, E. (2000). Predicting the performance of rescheduling strategies for parallel machine systems. *Journal of Manufacturing Systems*, *19*(4), 256–266. https://doi.org/10.1016/S0278-6125(01)80005-4

Vieira, G. E., Herrmann, J. W., & Lin, E. (2003). Rescheduling manufacturing systems: A framework of strategies, policies, and methods. *Journal of Scheduling*, *6*(1), 39–62. https://doi.org/10.1023/A:1022235519958/METRICS

Wang, M., Zhang, P., Zheng, P., He, J., Zhang, J., & Bao, J. (2020). An Improved Genetic Algorithm with Local Search for Dynamic Job Shop Scheduling Problem. *IEEE International Conference on Automation Science and Engineering*, *2020-August*, 766–771. https://doi.org/10.1109/CASE48305.2020.9216737

Wiendahl, H. P., & Garlichs, D. I. (1994). Decentral Production Scheduling of Assembly Systems with Genetic Algorithm. *CIRP Annals*, *43*(1), 389–395. https://doi.org/10.1016/S0007-8506(07)62237-9

Wojakowski, P., & Warzolek, D. (2014). The classification of scheduling problems under production uncertainty. *Research in Logistics & Production*, *4*(3), 245–256. https://bibliotekanauki.pl/articles/409534.pdf

Wu, S., Liu, E., Cao, R., & Bai, Q. (2025). Airline recovery problem under disruptions: A review. *Computers & Operations Research*, *175*, 106915. https://doi.org/10.1016/J.COR.2024.106915

Yan, J., Zhao, T., Zhang, T., Chu, H., Yang, C., & Zhang, Y. (2024). A Dynamic Scheduling Method Combining Iterative Optimization and Deep Reinforcement Learning to Solve Sudden Disturbance Events in a Flexible Manufacturing Process. *Mathematics 2025, Vol. 13, Page 4*, *13*(1), 4. https://doi.org/10.3390/MATH13010004

Zhang, B., Xiao, L., & Zhao, H. (2023). Rescheduling on Unrelated Parallel Machines with Rush Order. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3627915.3627922

Zhang, H., & Zhang, Y. Q. (2020). A discrete job-shop scheduling algorithm based on improved genetic algorithm. *International Journal of Simulation Modelling*, *19*(3), 517–528. https://doi.org/10.2507/IJSIMM19-3-CO14

Zheng, F., & Sui, Y. (2019). Bi-objective Optimization of Multiple-route Job Shop Scheduling with Route Cost. *IFAC-PapersOnLine*, *52*(13), 881–886. https://doi.org/10.1016/J.IFACOL.2019.11.241

Zhou, B., & Lei, Y. (2021). Bi-objective grey wolf optimization algorithm combined Levy flight mechanism for the FMC green scheduling problem. *Applied Soft Computing*, *111*, 107717. https://doi.org/10.1016/J.ASOC.2021.107717

Appendices

A Scenario overview

Table 1 and Table 3 provide an overview of the scenarios used in the experiments, as elaborated upon in Chapter 5. The scenario numbers correspond with the scenario numbers used in that chapter. The column 'Index' refers to the first position in the sequence of the initial schedule from which the schedule is disrupted.

Scenario	Index	Type	Duration [hrs]
0	2794	Execution delay	8.33
1	2466	Operator illness, Emergency order	48.64, 0.00
2	1597	Machine Failure	95.48
3	771	Material unavailability	12.21
4	1846	Operator illness	51.36
5	703	Material unavailability	11.84
6	899	Operator illness	52.52
7	946	Material unavailability	9.21
8	1857	Machine Failure	135.15
9	1607	Execution delay	13.51
10	503	Execution delay	4.94
11	2307	Material unavailability	10.07
12	857	Material unavailability	9.49
13	1852	Execution delay	7.04
14	348	Operator illness	41.12
15	1107	Machine Failure, Execution delay	87.33, 10.21
16	835	Material unavailability	9.67
17	2632	Emergency order	0.00
18	1511	Material unavailability	13.27
19	573	Material unavailability	11.29
20	1515	Execution delay	9.44
21	620	Machine Failure	70.90
22	2572	Operator illness	50.19
23	2624	Operator illness	54.40
24	1712	Machine Failure	72.21
25	1108	Operator illness	52.18
26	2488	Execution delay	13.77
27	99	Machine Failure	91.89
28	1157	Machine Failure, Emergency order	73.27, 0.00
29	151	Material unavailability	10.47

Table 1: Disruption scenarios with the type and duration for machine 538

Scenario	Index	Type	Duration [hrs]
0	298	Emergency order	0.00
1	1661	Operator illness	52.82
2	1986	Operator illness	51.08
3	680	Operator illness	45.07
4	1562	Machine Failure	76.46
5	2110	Operator illness	49.54
6	1039	Execution delay	12.01
7	1419	Machine Failure	108.77
8	476	Operator illness	40.94
9	472	Execution delay	15.67
10	546	Material unavailability	14.12
11	1051	Execution delay	14.69
12	519	Emergency order	0.00
13	69	Emergency order	0.00
14	1423	Execution delay	8.41
15	1422	Operator illness	57.73
16	2206	Operator illness	51.88
17	1366	Operator illness	48.4
18	560	Material unavailability	9.20
19	227	Operator illness	45.93
20	838	Execution delay	6.94
21	1681	Machine failure	56.12
22	917	Operator illness, Execution delay	47.0, 8.47
23	428	Execution delay	13.65
24	1622	Execution delay	13.21
25	173	Execution delay	9.80
26	1498	Material unavailability	6.64
27	1381	Execution delay	9.59
28	1652	Execution delay	8.09
29	368	Execution delay	7.32

Table 2: Disruption scenarios with the type and duration for machine 539

Scenario	Index	Type	Duration [hrs]
0	105	Operator illness	45.59
1	222	Material unavailability	7.21
2	91	Material unavailability	7.71
3	67	Material unavailability	9.06
4	43	Machine Failure	53.62
5	285	Material unavailability	11.34
6	39	Material unavailability	8.50
7	70	Execution delay	12.05
8	139	Operator illness	55.35
9	341	Operator illness, Emergency order	50.69
10	8	Machine failure	52.88
11	114	Machine failure	74.93
12	218	Execution delay	6.98
13	19	Emergency order	0.00
14	119	Execution delay	10.52
15	102	Material unavailability	10.72
16	284	Material unavailability	9.28
17	166	Material unavailability	10.34
18	333	Operator illness	48.55
19	229	Machine failure	112.08
20	96	Operator illness	52.24
21	203	Execution delay	10.97
22	346	Operator illness	46.63
23	33	Material unavailability	7.55
24	344	Machine failure	94.88
25	325	Material unavailability	6.27
26	22	Operator illness	49.10
27	167	Machine failure	93.84
28	26	Machine failure	53.71
29	54	Operator illness	45.91

Table 3: Disruption scenarios with the type and duration for machine 533

B Prototype interface

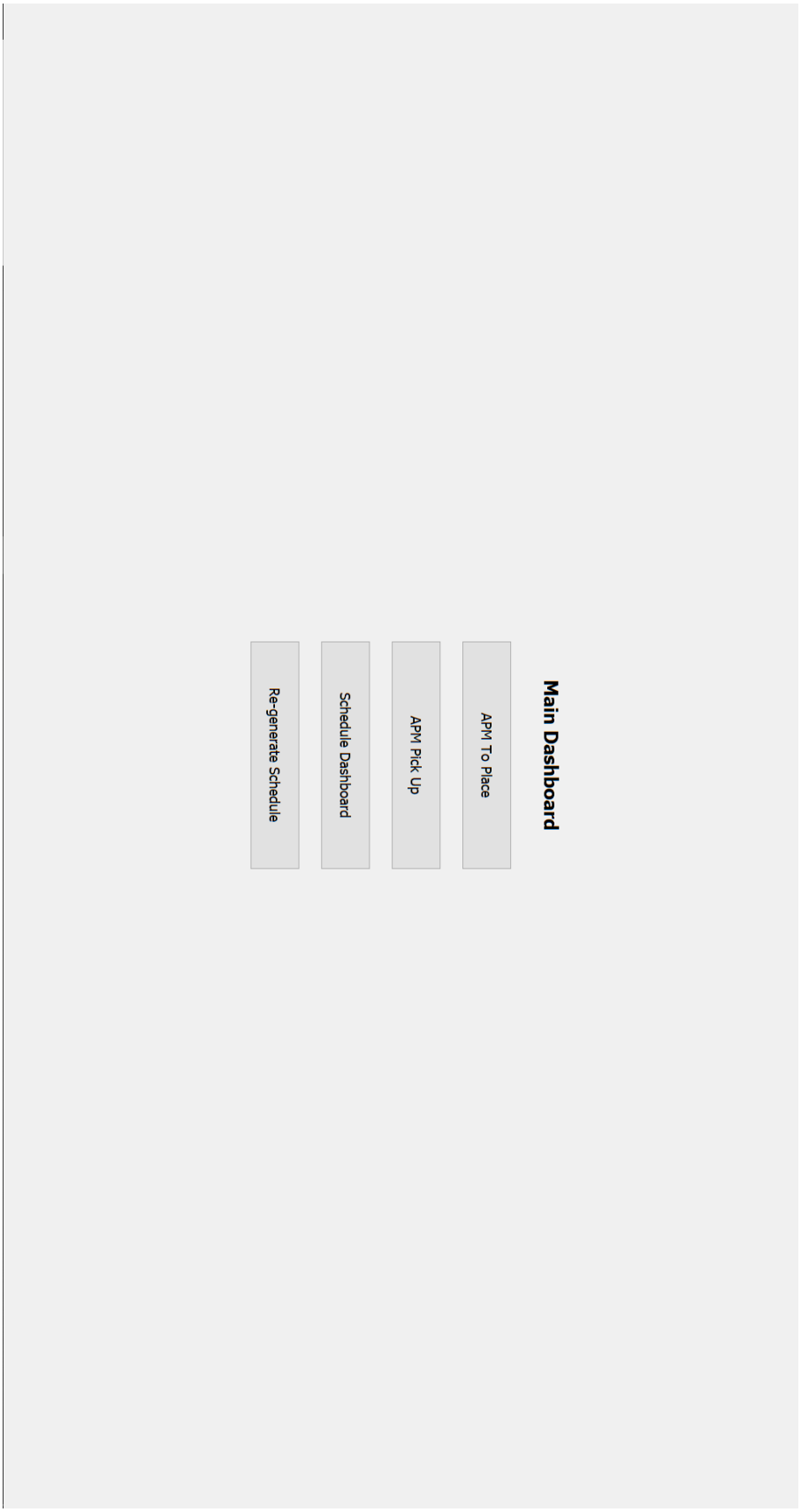


Figure 1: Main dashboard page of the prototype developed by the Fraunhofer Innovation Platform