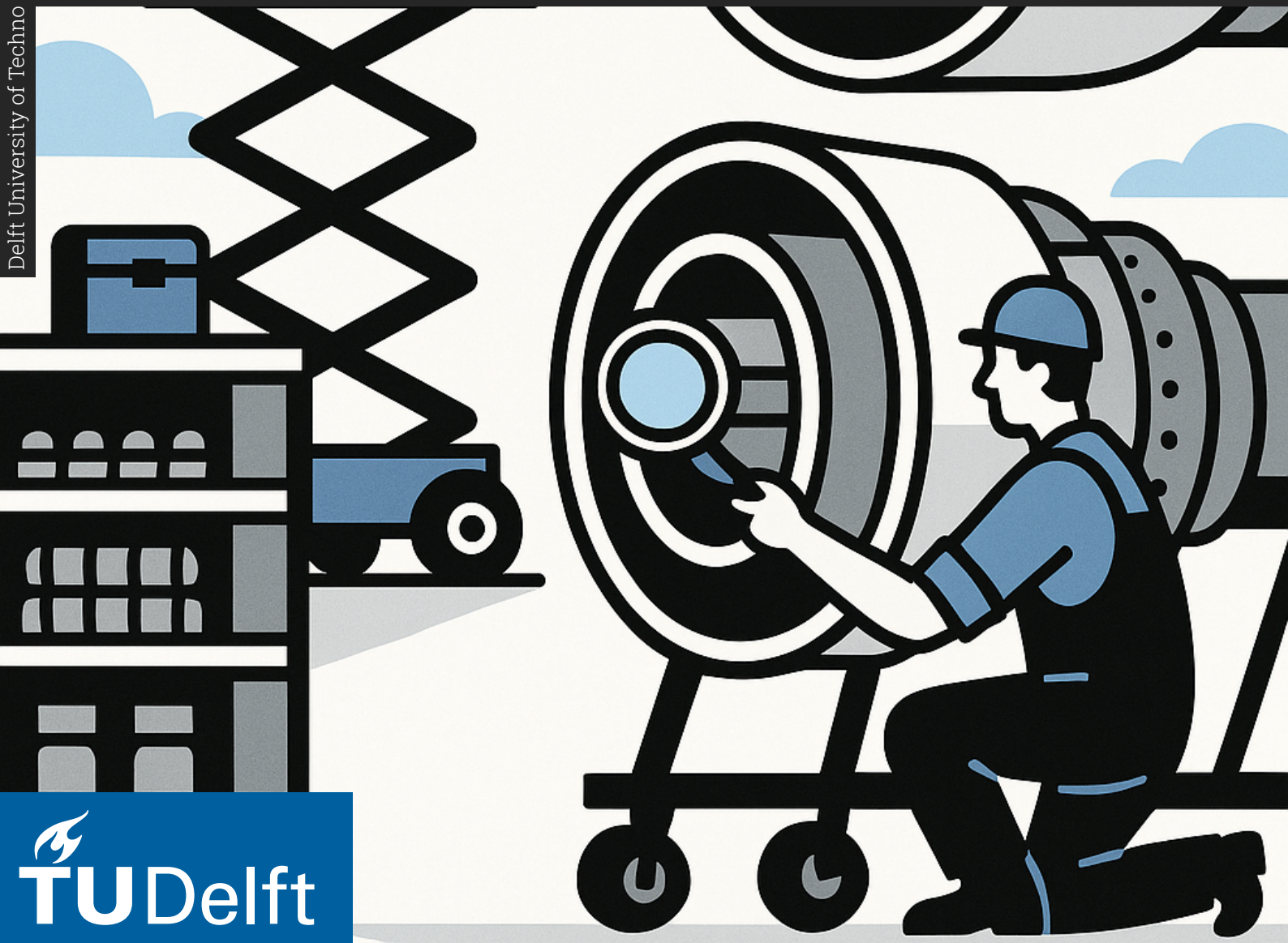
Proactive–Reactive Rescheduling for RCMPSP/max using Exact Methods

Optimisation of disruption-prone schedules of an aircraft engine MRO

Jasper Klein Kranenbarg

Delft University of Techno



TU Delft

Proactive–Reactive Rescheduling for RCMPSP/max using Exact Methods

Optimisation of disruption-prone schedules of an aircraft engine MRO

by

Jasper Klein Kranenbarg

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number: 4586387
Project duration: December, 2024 – Oktober, 2025
Thesis committee:

Dr. N. Yorke-Smith,	TU Delft, supervisor
Dr. Ir. B. Lopes Dos Santos,	KLM, external supervisor
Dr. M.J. Ribeiro,	TU Delft, independent examiner

This thesis is confidential and cannot be made public until 16-10-2026

Cover: AI-generated image of a typical aeroplane engine MRO

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

As my studies at the TU Delft finally come to an end, I would like to take a moment to express my gratitude to everyone who helped me along this, *slightly longer than intended*, journey.

I want to start by thanking Neil, Bruno and Serge for their guidance and support along the way. Although my thesis did not always go according to plan, without your guidance, insights and critical questions on my work, I would not have been able to make it this far.

I would also like to express my gratitude to the people at KLM who helped me along the way. Starting with the other people from the DT&CI team, for helping me get in touch with the people I needed, brainstorming on any issues I encountered and, most importantly, for making me feel part of the team and for the many good times during my thesis internship. On top of this, I would like to thank Thijs for taking the time to help me explain the current processes and for providing me with the data I needed whenever possible.

Next, I want to thank my parents and sisters for their unwavering support since I began my studies; without them, I wouldn't have made it this far.

And finally, I would like to thank Manon for supporting *(and putting up with)* me during the thesis.

Jasper Klein Kranenbarg
The Hague, October 2025

Summary

Over the last few decades, the number of people flying has grown significantly [15][16]. However, during the pandemic, this number dropped significantly. Now, a few years since the pandemic ended, air traffic is back to the level it was before the pandemic, and is expected to grow even faster than before the pandemic [9]

Although this might seem like good news to any aircraft operator, does this also require those same carriers to have more aeroplanes available to them at any time? According to the IATA [8], the pandemic's impact is still clearly evident. The supply of new engines and replacement parts falls short of meeting global demand. Focusing on parts specifically, even if one can place and receive an order for replacement parts, does this not provide any guarantee regarding the number of parts that can be ordered, when those parts will be delivered, and when the next order will be fulfilled? Even if one were able to order these parts regularly, most aircraft engine repair facilities still face personnel shortages.

Most aircraft engine repairs are organised as a "Maintenance, Repair and Overhaul" (MRO). Scheduling projects for an aircraft engine MRO presents several key challenges. The first challenge refers to the specific tooling. It is used in the various stages of the disassembly and assembly process, which are not only limited but also often used for both disassembly and assembly of any component (i.e. engine stands). The second challenge is about the uncertainty of processes (especially when depending on external parties). As mentioned earlier, at the time of writing, the supply chain of these MROs remains heavily disrupted, resulting in several parts being (near) unobtainable. The final main challenge involves the personnel. Even if all other resources are ready and available, this means nothing if there are not enough personnel with the required skills available.

The goal of this research is to develop a model that uses exact methods to perform proactive rescheduling of the aeroplane engine MRO process. This model focuses on finding a solution that balances between: (1) **Quality**, (2) **Deviation impact minimisation** and (3) **Robustness**. Quality refers to metrics such as project turnaround time (TAT), workload performance, and minimisation of unused workforces. Robustness metrics focus on aspects such as minimising the number of excessively scheduled resources in a solution or indicating how well a solution responds to disruptions in the input data. Finally, deviation impact minimisation focuses on minimising the impact reschedules have on the actual workflow of the MRO. It is undesirable to deviate from any jobs that are already in progress or planned to start in the near future; thus, the model aims to perform most deviations outside this "near future" scope, unless doing so would outweigh a minimal deviation in the near future.

Contents

Preface i

Summary ii

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Problem description . . . 1
  - 1.2.1 Problem objective definitions . . . 2
- 1.3 Evaluating schedules . . . 2
- 1.4 Research Question(s) . . . 3
- 1.5 Research goals . . . 3
- 1.6 Input data limitations . . . 3
  - 1.6.1 Ideal input data . . . 3
  - 1.6.2 Historical data . . . 4
  - 1.6.3 Description of the environment . . . 4
  - 1.6.4 Project profiles . . . 4
- 1.7 Contributions . . . 4

2 Related work 5
- 2.1 Resource constrained scheduling problems . . . 5
  - 2.1.1 RCPSP . . . 5
  - 2.1.2 RCPSP with time-varying capacities . . . 5
  - 2.1.3 RCMPSP . . . 6
  - 2.1.4 RCPSP/max . . . 6
- 2.2 Modelling techniques for (re-)scheduling problems . . . 6
  - 2.2.1 Heuristic methods (non-exhaustive search) . . . 6
  - 2.2.2 Exact methods . . . 7
  - 2.2.3 Heuristic vs Exact . . . 8
- 2.3 Robust scheduling . . . 8
  - 2.3.1 Measuring robustness . . . 8
  - 2.3.2 Robust CP scheduling . . . 9
- 2.4 Aircraft engine MRO scheduling . . . 9

3 Methodology 10
- 3.1 Experimental setup . . . 10
  - 3.1.1 Input datasets . . . 10
  - 3.1.2 Solver selection . . . 11
- 3.2 Reschedule model evaluation metrics . . . 11
  - 3.2.1 Quality evaluation metric . . . 11
  - 3.2.2 Robustness evaluation metric . . . 11

4 Proposed model 14
- 4.1 Model definition . . . 14
  - 4.1.1 Notation . . . 14
  - 4.1.2 Constraints . . . 16
  - 4.1.3 Objective Function . . . 17
  - 4.1.4 Objectives . . . 18
- 4.2 Model description . . . 18
  - 4.2.1 Experimental setup . . . 18
  - 4.2.2 Implementation . . . 18
  - 4.2.3 Simulation . . . 19

5 Experimental evaluation **20**
5.1 Environment modelling and schedule evaluation . . . 20
5.1.1 Optimised and Reference model descriptions . . . 20
5.1.2 Method . . . 20
5.1.3 Insights from solving . . . 23
5.1.4 Performance evaluation . . . 23
5.2 Robustness objectives in CP modelling . . . 24
5.2.1 Robustness metrics . . . 24
5.2.2 Impact of robustness constraints . . . 24
5.3 Evaluation under disruptions . . . 25
5.3.1 Simulating disruption scenarios . . . 25
5.3.2 Robustness evaluation . . . 26

6 Conclusion **29**
6.1 Conclusion . . . 29
6.2 Discussion . . . 29
6.2.1 Data quality and availability . . . 29
6.2.2 Model Enhancements . . . 30
6.2.3 Reliability of results . . . 30
6.3 Recommendations . . . 30

References **31**

A Example of the rolling time window for the fever score **33**

B Metrics for various combinations of hyperparameters **34**

C Full comparison of model statistics during simulations introducing disruptions **36**

1

Introduction

1.1. Context

Over the last decade, demand for aerial travel has increased significantly [15]. However, in that same decade, more and more people are aware of the environmental impact caused by aeroplanes. Various methods to meet the two conflicting trends in the middle include developing more sustainable engines that emit fewer greenhouse gases, produce less noise, or are even completely carbon neutral.

Due to these needs, newer engines are becoming increasingly efficient, and engines that utilise different types of fuel are being researched and developed. This would also cause engine maintenance to change along. At the same time, the engines currently widely used are all composed of modules of a similar type.

At one of Europe's largest aircraft engine MROs, the aircraft engine inspection cycle is divided into five stages (called gates 0 to 4). The first and final gates are related to the arrival and departure of the engine from the shop. The other gates present the disassembly, repair and reassembly processes, respectively.

Engines can enter the shop for two main reasons: scheduled maintenance and repairs. During scheduled maintenance, the scope of repair work is typically known in advance. It includes tasks such as replacing parts with a predetermined limited lifetime (for example, rubber rings that must be replaced after a certain number of rotations). On the other hand, repairs are addressed when specific problems arise. For instance, if an engine suffers a bird strike, the exact damage can only be assessed by disassembling the engine to identify the issues.

Every engine enters the shop at gate 0 and moves to gate 1 once the repairs/inspections begin. At gate 1, the engine is disassembled and inspected. Then, based on their findings, the parts/modules that did not pass the inspection are sent to Gate 2. At gate two, these parts/modules are either repaired (on-site or by a third party) or replaced with new parts that must be ordered. Then, once all these parts/modules have been restored, the reassembly process begins at gate 3, led by the same team of mechanics that disassembled the engine at gate 1. Once that is done, the engine goes to gate 4 and departs the engine shop.

1.2. Problem description

In environments that are inherently prone to disruptions, such as those involving aircraft engine maintenance, repair, and overhaul (MRO), operations are susceptible to various interruptions. These disruptions can be internal, like employee sickness, broken tools, and maintenance issues, as well as external, including delays in repairs, late orders, personnel shortages, and global resource limitations, among others.

As can be seen, most of the external disruptions can be traced back to (global) shortages. According to the *International Air Transport Association*(IATA), demand for aircraft has not only grown over the last

decade. Still, it has also increased at a faster rate than the production growth. Furthermore, the global backlog of aircraft maintenance has also been growing [8], making it more crucial than ever to create the most effective schedules. Since this problem is not expected to be resolved in the coming years [7], this can lead to problems in other areas. According to Reuters [5], airlines are delaying the retirement of older planes (less efficient planes) simply because newer plane models are not available. And even if an airline manages to acquire a newer model, there is no guarantee that repairs and replacement parts will be readily available.

The MRO environment used for the thesis has three levels of scheduling. The first level schedules years in advance and is primarily used by the sales department to inform them of the types of projects that can be undertaken at specific times. The second level is months in advance and defines which weeks particular projects should be inducted or leave the shop. The final level, referred to as the **short-term planning**, which is the focus of this thesis, spans the next three weeks. This final level is where the thesis focuses; in this level, the jobs of each project are scheduled per shift.

This research aims to improve the overall productivity of the MRO, thereby maximising the utilisation of available resources.

1.2.1. Problem objective definitions

The problem posed in this research has multiple objectives, which are **Quality**, **Deviation** and **Robustness**. When talking about the quality of a solution, it refers to the current (financial) value of the solution. For example, TAT, idle authorisations per shift and performed workload.

Deviation of a solution refers to penalties based on whether the changes were unfavourable. For instance, delaying a project that is planned to start two weeks from now is not a significant issue. However, the nearer the task is to the time horizon, the higher the penalty would be. It is also possible to deviate from the duration of a shift, for example, by rushing a job or experiencing a disruption during an active job. This is why deviation penalties based on start/end time are only counted once, while penalties based on changes in job durations are counted at every step of the simulation.

Finally, schedule robustness refers to a solution's ability to handle disruptions effectively.

1.3. Evaluating schedules

In this work, schedules are evaluated on three metrics; more extensive definitions can be found in section 3.2.2. These metrics are **quality**, **deviation** and **robustness**. When discussing schedule quality, it concerns every metric related to how well the environment is expected to perform based on the view from a single point in time. Deviation concerns everything related to the costs of rescheduling. And finally, robustness, which is used to indicate how well a schedule is able to handle unexpected disruptions in the environment.

Optimising for these three metrics often leads to some trade-offs between them. By prioritising quality, it can likely cause numerous and significant changes to the schedule, as well as compromise robustness by aiming to schedule as tightly as possible. Prioritising minimal deviations makes the schedule less dynamic and prevents actions after disruptions or when new projects are inducted. If robustness is prioritised, it would likely lead to a spread-out schedule, meaning less quality and potentially significant deviations after disruptions (because it would not only utilise the robustness but also look to regain this robustness afterwards).

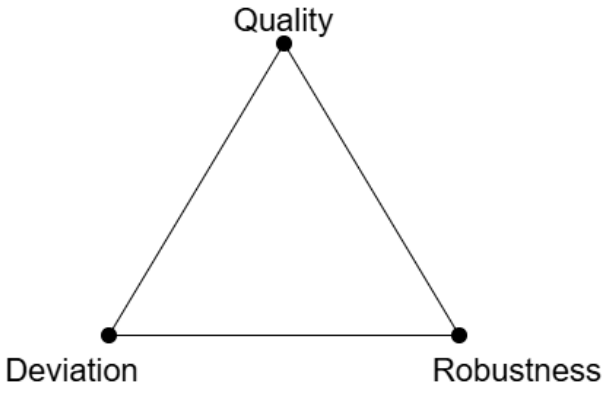


Figure 1.1: Overview of different objectives

1.4. Research Question(s)

Currently, the schedulers for the short-term plan-
ning use a planning tool that helps them plan for the short term. However, this tool does not help with optimising this planning, nor does it assist with rescheduling after disruptions take place. This led me to the following research question:

> How can constraint programming facilitate proactive and reactive rescheduling for an RCMP-SP/max formulation for an aircraft engine MRO, while improving schedule quality, limiting reschedule deviation, and enhancing robustness within short solve times?

The goal is to demonstrate how scheduling using CP facilitates the creation of more robust schedules in advance, as well as how CP can aid in rescheduling after disruptions to achieve a more optimal schedule.

Which I aim to answer using the following sub-questions:

1. How can an aircraft engine MRO environment be effectively modelled and analysed as an RCMP-SP/max framework?
2. How do robust schedules perform against regular schedules when no disruptions occurs?
3. How do robust schedules perform compared to the reference schedules when simulating disruptions?

The goal is to start by deriving a model that finds solutions quickly (it must be usable in real-time; if the model will be used in an actual MRO, it should be able to provide new schedules within two minutes). This model starts by focusing only on quality and deviation (to make sure new schedules are based on previous schedules).

Once this model has been defined and implemented, additional constraints are added to the model that aim to improve schedule stability, flexibility and quality robustness. The impact of these added constraints is then verified by comparing two instances with equal settings, except that one has robustness added to the objective, while the other does not. During these simulations, no disruptions are added yet. It is expected that schedules that also focus on robustness will have a slightly worse quality than those that do not focus on robustness.

However, by evaluating the quality of reference schedules against robust schedules when disruptions do occur, it would indicate whether the initial quality loss is worth the reduced quality loss after disruptions.

1.5. Research goals

The primary objective of this research is to assess the effectiveness of exact methods in proactive rescheduling within such an environment. If it can be shown that exact methods can be used for this purpose, it would be possible to guarantee some level of optimality (unlike heuristic methods).

1.6. Input data limitations

The initial purpose of the research results was to assist schedulers in managing disruptions by providing a solution that enables reactive rescheduling in response to disruptions. However, during the research period, it became increasingly clear that this would not be realistic with the current state of data availability and documentation of the environment. At the time of writing, significant efforts are underway in that environment to resolve these limitations, which is why this section is included. To provide insights as to why certain decisions were made.

1.6.1. Ideal input data

To introduce, a short explanation of what input data would be available in the ideal scenario.

First of all, the availability of historical data. This would ensure that all changes in the data are logged appropriately. Both any reasons for changes as well as any adjustments or initial planning. This would allow for thorough historical analysis, as well as enabling simulations.

Secondly, a complete description of the environment. This includes, at a minimum, the capacities of every resource, as well as any setup times.

Finally, an extensive description is available for all types of projects. This includes all jobs, dependencies between jobs and any additional information specific to the project (contract costs, TAT, …). Additionally, for each job, a specification is provided on the required resources, time/fte, and required authorisations.

Unfortunately, the available data falls far short of this ideal scenario.

1.6.2. Historical data

Until recently, the short-term schedule was maintained using a shared Excel spreadsheet. Every planner would make adjustments their own way. The diversity of how these changes were done made it nearly impossible to process automatically. Additionally, if there were changes in the schedule, there was no standard method for noting them. Different types of notations were, for example:

- Filling cells with a red colour.
- Adding text to one of the cells of the disrupted task.
- Adding a remark at some related cell (header, project information or the actual task).

After numerous failed attempts at accurately parsing these Excel sheets, they were excluded from the input data.

However, a few weeks ago, a transition to a planning tool called Anago[1] was initiated. The consistency of the data exported from Anago is significantly improved; however, it is not perfect, especially in the earliest exports, which often included human errors when entering projects.

Luckily, it was easy to identify these errors, and exports with mostly erroneous data were excluded from the input data.

1.6.3. Description of the environment

Another hurdle was providing descriptions of the environment, especially regarding workstations and tooling. Before Anago's export became available, I could barely find any files or documents on what stands and tooling are used at various stages of the process, and what authorisations are required. Then, when these exports became available, parts of the data gaps were filled by definitions on the capacity of the engine stands (partially), as well as their capacity

1.6.4. Project profiles

Project profiles serve as the foundation for each type of project. These profiles define the duration of each sub-job and the resources it consumes. The only problem is that most profiles are outdated or lack critical information, like the amount of work each sub-job requires

1.7. Contributions

This work provides the following contributions:

- Provide a performance evaluation of rescheduling the RCMPSP/max instance using constraint programming.
- Provide an overview of the expected impact robust rescheduling has on the quality of a schedule when compared to rescheduling towards minimal changes and maximal quality when no disruptions take place.
- Provide the results of a simulation based on historical data that introduces random sets of disruptions. The simulations are always performed for both methods, and a comparison is done on instances experiencing the same sets of disruptions.

2

Related work

We begin by providing some additional information on topic related to this work, as well as briefly a related work

2.1. Resource constrained scheduling problems

When scheduling for an environment where jobs that use shared (limited) resources have to be planned, then this is usually some form of the **Resource-Constrained Scheduling Problem** (RCPSP)[3]. Countless variations of RCPSP exist, each with its own unique properties. For example, the posed MRO scheduling problem can be defined as an **Resource-Constrained Multi-Project Scheduling Problem with minimal and maximal time lags** (RCMPSP/max). This describes a scheduling problem where multiple projects co-occur, utilising the same limited resources, and the timing of task orders can be dynamic. This means that dependencies between jobs are not strictly limited by the end time of another job, but that the dependencies are defined by certain bounds in which jobs can begin.

This section describes RCPSP and several extensions to the problem relevant to the thesis. For each of these problems, the same discrete time definition t is used. Which is defined as $\forall t \rightarrow t \in \{0, 1, \ldots, H\}$ given time horizon of length H. In later parts of this thesis, these time steps will also be referred to as shifts, corresponding to the definition of time steps in the context.

2.1.1. RCPSP

RCPSP problems are made up of three main components: a project P, a set of renewable resources $\mathcal{K}$ and a set of temporal constraints E. The definition of this *regular* RCPSP is as follows:

Given a project P that consists of a set of activities (also referred to as jobs) $\mathcal{J}$. Each activity $j \in \mathcal{J}$ has a defined start S_j, end C_j and duration d_j. For which holds that $\forall j \in \mathcal{J} \rightarrow \ S_j + d_j = C_j$.

The set $\mathcal{K}$ refers to the set of renewable resources. Each resource k has a defined capacity function $R_k(t)$ which represents how much of the resource is available at time t. Note that in regular RCPSP, the capacity of each resource remains constant at all times.

Finally, the set E defines the precedence constraints between the jobs in P. For regular RCPSP, are precedence constraints defined as **Finish-Start (FS)** precedences. This means that given jobs $j_a, j_b \in \mathcal{J}$ where j_b depends on j_a, the precedence constraint is defined as $C_{j_a} \leq S_{j_b}$.

2.1.2. RCPSP with time-varying capacities

The first extension relevant to the thesis is the version that uses variable resource capacities instead of constant capacities. **RCPSP with time-varying capacities** uses nearly the exact definition as regular RCPSP, except that the capacity function $R_k(t)$ no longer returns a constant variable, but is now able to return different values for different moments in time.

2.1.3. RCMPSP

The **resource constrained multiple projects scheduling problem** is another extension to RCPSP that focuses on scheduling multiple projects instead of a single project, while using the same environment and sharing resources. In this variant of RCPSP, instead of a single project P, it is now defined by N projects $\mathcal{P} = P_0, P_1, \ldots, P_N$. This not only impacts the complexity of the problem, but it also affects the definition by introducing **project release times** and optional **project deadlines**.

The project release time for a project P is defined by $\mathcal{R}_P$ and corresponds to the moment in time when the project can be started. Optionally, a project deadline can also be defined, which sets an upper bound to when a project should be finished.

This extension is significantly harder, as resources are shared and utilisation is critical at specific times.

2.1.4. RCPSP/max

The resource-constrained project scheduling problem with minimal and maximal time lags (RCPSP/max) is another extension to classic RCPSP. Whereas RCPSP makes use of the mentioned FS precedence constraints, RCPSP/max makes use of four adaptations of the FS precedence constraint:

- $FS^L \rightarrow$ Finish-Start with time lags
- $SS^L \rightarrow$ Start-Start with time lags
- $FF^L \rightarrow$ Finish-Finish with time lags
- $SF^L \rightarrow$ Start-Finish with time lags

Whereas FS precedences can only constrain the order by defining which jobs should finish before which other jobs can start, are time lag precedences able to describe windows in which the succeeding start/finish event should take place (relative to the corresponding preceding start/finish event). For each time lag L, it holds that $L = \{L^{min}, L^{max}\}$, $L^{min} \leq L^{max}$ and $L^{min}, L^{max} \in \mathbb{Z}$. Which expands the precedence constraint $FS = C_{j_a} \leq S_{j_b}$ to:

- $FS^L = C_{j_a} + L^{min} \leq S_{j_b} \leq C_{j_a} + L^{max}$.
- $SS^L = S_{j_a} + L^{min} \leq S_{j_b} \leq S_{j_a} + L^{max}$.
- $FF^L = C_{j_a} + L^{min} \leq C_{j_b} \leq C_{j_a} + L^{max}$.
- $SF^L = S_{j_a} + L^{min} \leq C_{j_b} \leq S_{j_a} + L^{max}$.

These four precedence constraints all define a window within which the succeeding event must occur relative to the time the preceding event occurred.

2.2. Modelling techniques for (re-)scheduling problems

Almost everything you can think of has to do with some scheduling problem, from the manufacturing of your belongings to the people stocking shelves in your grocery store. However, each of these problems has its own unique properties and requirements. Some problems focus on finding optimal schedules months in advance, while others focus on scheduling in real-time based on events. As one can imagine, there is no single best modelling technique used for all scheduling problems. The most commonly used techniques can mainly be divided into **heuristic methods** and **exact methods**. Heuristic methods are typically employed when a problem is too complex or requires a solution that must be achieved as quickly as possible. Exact methods are commonly used in less complex scheduling problems or when a high level of optimality is required.

This section describes various processes commonly used to solve (re-)scheduling problems. First, several heuristic methods are outlined, followed by the most widely used exact methods (constraint programming (CP) and mixed integer programming (MIP)). For each technique, the strengths and weaknesses are discussed in the context of the thesis.

2.2.1. Heuristic methods (non-exhaustive search)

Heuristic approaches for RCPSP and variants construct or improve schedules quickly; however, this is done without providing any optimality guarantees.

The two most common heuristic methods for solving RCPSP instances are priority-rule procedures with serial/parallel schedule-generation schemes (SGS) that dispatch activities by rules such as earliest finish time or minimum slack [13]. The second type of commonly used heuristic methods is those based on metaheuristics, such as tabu search, simulated annealing, and genetic algorithms (GAs). These methods perform a search over activity lists and have to enforce feasibility via an SGS or by performing a repair. [4]. Performance of these methods usually requires specific tuning to the problem at hand, and even then, these methods are unable to provide any proof of optimality.

Local search

Local search methods *traverse* the search space to find the optimal solution. The most commonly used local search methods are **Genetic Algorithms (GAs)**.

Genetic Algorithms explore the search space by simulating evolution. The basic GA, which serves as the basis for all variants, works as follows. Start by defining an initial population P_0 of N candidate solutions, then, until some termination criterion is met:

1. Select pairs of parents from P.
2. Create offspring population O by performing crossover between the parent solutions.
3. (Optional) Mutate offspring solutions randomly.
4. Evaluate offspring solutions.
5. Replace the population P by O.

Priority rules

Priority-rule dispatching methods work fairly simple. Each project is assigned a priority value. During schedule generation, whenever a new project can begin, choose the project with the highest priority.

2.2.2. Exact methods

Whereas heuristic methods perform a non-exhaustive search by exploring the search space and seeking optima, exact methods conduct an exhaustive search by pruning the search space until the optimal solution is found. One of the most common exact methods used to solve RCPSP instances is through constraint programming.

Constraint programming (CP)

CP is a declarative language that can be used to model and solve problems by defining decision variables, constraints and an objective function. Decision variables can be set to any value within their domain, constraints are used to define infeasible combinations of value assignments to the decision variables, and the objective function describes the goal.

Multi-objective CP

One problem with CP is that there is no single way to optimise towards multiple objectives. A model is always defined by an objective function that determines the quality of a solution. When multiple objectives matter, the objective function for a multi-objective problem with N is usually composed something like: $Z = Z^0 + Z^1 + \cdots + Z^N$. This means we cannot simply let the model optimise towards the best value of an objective, such as A, and then B, and so on. There are several ways to overcome this limitation; the most commonly used methods are **Weighted Sum (WS)** and **Relax-and-Solve (RS)**.

The weighted sum assigns a weight value to each sub-objective, like $Z = w_0 * Z^0 + w_1 * Z^1 + \cdots + w_N * Z^N$. The advantage of the weighted sum is that it is capable of optimising towards each sub-objective simultaneously in a single iteration. However, if the domains of the various sub-objectives are too large and/or different, finding suitable weights might be challenging.

Relax-and-Solve overcomes this issue by solving the problem multiple times. This approach works by solving a sub-objective, using the solution to set a bound on the objective, and then solving it for the next sub-objective until all objectives are met. Unlike what one would expect, running the model multiple times does not speed up solving the problem in specific scenarios. For example, given a complex scheduling problem with different resources with varying consumptions. First, solving for just

finding the fastest schedule and then finding the optimal resource distribution using a bound on the previous sub-objective.

Rescheduling using CP

Complex rescheduling problems are typically solved using heuristic methods. However, the main difference between heuristic and exact methods is that heuristic search is not exhaustive, meaning it can become stuck at local optima. A CP approach offers a thorough search; however, since the problem is NP-hard, it requires some additional methods in the model to reduce the search space.

Then, specifically regarding rescheduling, the most common way it is performed in CP is by adding a deviation component to the objective function. This *deviation* component is based on the extent to which the found schedule deviates from the previous schedule.

2.2.3. Heuristic vs Exact

I decided to use an exact method to solve the posed problem in the thesis, primarily due to the potential optimality guarantee and how it reschedules a rolling time window.

Potential optimality guarantees are crucial for the goal it has to achieve. Especially after a disruption has occurred, a heuristic method seeks a new optimum in the vicinity. Whereas the exact process can look for a solution of at least the same quality as the previous solution. It is likely to quickly determine whether there is still a feasible solution of at least the same quality as the previous solution.

Rescheduling a rolling time window is expected to be the reason for any rescheduling that has to take place. The exact method is expected to outperform heuristic methods. Because there are likely to be little to no changes in the first parts of the rolling time window, by starting the search at the new variables in the environment using the last solution as an initial value for all other variables, it should be able to prune a lot of the search space, as well as likely start the search near the optimal solution using the value assignments of the optimal solution for most variables. Also, suppose at the next step of the time window, a new project is announced to start at a time where it conflicts with other already known projects on some resources it requires. A CP solver will likely be able to identify which conflicts are conflicting quickly. This should cause the solver to start by searching for the optimal solution, focusing only on the affected parts of the search space. This would be a lot harder when using heuristic methods, since you would need some way to not only find out which areas of the search space are affected, but you would also need to steer the heuristic method in such a way that it would only focus on the affected parts.

2.3. Robust scheduling

The goal of robust scheduling is to find solutions that not only have high quality but are also able to retain that quality after disruptions.

2.3.1. Measuring robustness

Robustness of a schedule refers to its ability to withstand unexpected disruptions and remain efficient. This can be measured by numerous quality metrics, the most common being: **stability** (solution robustness), **quality robustness** and **flexibility** (cost of schedule repairs).[6]

Solution robustness

The solution robustness metric indicates a solution's stability. If a solution has high stability, most disruptions would result in small to no changes to the existing schedule. If a schedule has low stability, even the most minor disruptions can necessitate significant adjustments to return it to a feasible state.

Quality robustness

The quality robustness metric helps determine how well a solution retains its quality in the face of disruptions. A low-quality robustness indicates that any disruptions are likely to impact the quality of the schedule. Where high-quality robustness means that the current solution quality remains a valid estimate of the solution quality after any disruptions occur.

Repair costs

The final robustness metric refers to the stability of a solution. Where a solution has low stability, any disruptions lead to large (processing) costs to find a new schedule. A high stability is achieved when rescheduling takes little time with few resources.

2.3.2. Robust CP scheduling

Although these metrics seem simple, the model is not able to optimise directly to them without exponentially increasing the time needed to solve the problem.

To resolve this issue, several solutions have been proposed in the past. These solutions primarily focus on two key objectives. The first goal is to minimise the maximum resource consumption at each point in time. The other goal focuses on minimising the number of jobs that are unable to move to a nearby moment in time (minimise the number of jobs with little to no temporal flexibility).

For the context described in this thesis, the second goal is less relevant than the first. Since there are jobs that are fixed to each other, as well as jobs with time lags, meaning some jobs are unable to utilise this buffer (such as sequential jobs where the engine remains idle), and some jobs are already defined to have dynamic start/end times.

The first goal, on the other hand, is much more critical. Since disruptions are expected to occur frequently, jobs will likely require resources for more extended periods or at different times. By minimising the number of capped out resources at each point in time, it is less likely that a disrupted job will impact other projects.

Section 3.2.2 shows how newly proposed *fever scores* are added to the model, a metric that is evaluated while solving and penalises high resource consumptions based on the remaining capacity.

2.4. Aircraft engine MRO scheduling

Recently, J. Cho [2] researched the feasibility of scheduling an aircraft engine MRO using CP. This work used historical data logs from Korean Air's aircraft engine MRO as input, and they designed a model that creates a schedule from the ground up for a fixed time horizon of one year. Although their results look promising for a similar environment to that described in this thesis, their primary goal was to demonstrate that CP was a feasible solution capable of providing high-quality schedules. This was evaluated based on the following metrics:

1. The number of days required to finish all the given work.
2. The number of workplaces required to finish all the given work in a year of 260 working days.
3. Resource utilisation—evaluated based on the optimal use of available workplaces.

Only the first performance metric would be suitable for the problem posed in this thesis, where the goal is to perform robust rescheduling of the short-term planning. The second metric is intended for resource management (which is rarely done in the short term) to find lower bounds for the required capacities of the various resources. The final metric is similar to the second metric, except this metric is used both to indicate excessive available capacities and to minimise unused resources.

The main issue with this paper concerns the authors ' view on handling robustness. In the introduction, the authors mention the numerous ways disruptions can occur in these environments. However, instead of discussing how to minimise the impact of these disruptions, the authors use them as a reason why efficient schedules are essential.

This thesis aims to fill this knowledge gap by proposing a robust predictive–reactive rescheduling method. This uses a rolling time window to reschedule whenever changes in the environment occur (disruptions, new projects, …). Instead of using the metrics proposed by Cho, the proposed model is used to describe the metrics in 3.2.2. Which correspond to schedule quality, reschedule impact and robustness accordingly.

3

Methodology

3.1. Experimental setup

3.1.1. Input datasets

As mentioned in section 1.6, not all required data was readily available. Which is why the input data is a combination of what was available and manually added definitions (based on information gained from talks to planners).

The exports from Anago were used as **reference schedules**. These reference schedules contain most of the data needed to simulate the MRO environment for the corresponding period.

The information extracted from each project was as follows:

- Engine Serial Number (ESN)
- Engine type (CFM57-B/CF6/LEAP/GEnx)
- Workscope (Major/Minor)
- Turnaround Time (TAT):
  - Contract TAT (Agreed maintenance duration in days)
  - Actual TAT (Number of days since induction)
- Administrative properties (Customer name, project name, etc.)
- Jobs (**Planned** jobs (numerous jobs are missing from these files))

Each job would be defined by:

- ESN (Project identifier)
- Job type
- Planned timing (potentially incomplete, if the start and/or end is outside the time window of the export)
  - Starting shift (when a job is scheduled to begin)
  - Completion shift (when a job is scheduled to be completed)
- Required number of FTE at each scheduled shift (not available for each job)

After merging the different reference schedule versions, each job of each project was validated and repaired. This was done using either reference project profiles estimated from available datasets or a fallback profile that checked whether all jobs were present and if all jobs met the basic temporal constraints used for each project. All projects that had an available corresponding reference project profile were then first **extended** and then **expanded**. Extending projects meant that all jobs of a project that were cut off by the export time window were extended to meet the reference duration. Expanding projects is nearly the same, except that it examines which jobs were likely to be completely cut off in

the reference schedule and adds these jobs using the duration and resource requirements from the reference profile.

The reference schedules also contain for each shift the corresponding **resource values** (availabilities and planned consumptions) for both authorisations and workstations. These **reference resources** frequently have more load assigned than available. This **excessive consumption** differs in severity. For workforce resources, excesses might not be as excessive as they seem, since the workforce requirements for each job are not as specific as they appear to be. This is why only the availability of the workforce is used for the model. However, for workstation resources, the only available resources are engine stands and the test cell, which is why excessive consumption in the reference schedule is an actual issue. However, the planners have decided that this should somehow be possible. This is why, for workstation resources, there are two definitions for capacity. **Documented capacity** refers to the available resource value of each workstation, and **scheduled capacity** refers to $max(available, assigned.$

Initially, both were used for determining penalties in the model, as excessive consumption should always be avoided; however, a fair comparison between the found solutions and the reference schedules led to nearly the same results for almost all instances using scheduled capacity. Since the scheduled capacity accounts for the exact resource shortage encountered in the reference schedule, the model often yields results that align precisely with the given scheduled capacity.

3.1.2. Solver selection

Next, a solver was chosen. As shown in Section 2.2.2, every solver works differently, meaning that there is no single solver that works best for all types of problems. Currently, the best options for RCMPSP/max are IBM's ILOG **CP-Optimiser**[10] and Google's **OR-Tools**[12].

Earlier this year, Lan and Berkhout published a study[11] where the performance of these two solvers was compared using several types of scheduling problems (including RCPSP, MMRCPSP and RCMPSP). Because the comparison shows that the solvers have similar performance, the decision was made to choose OR-Tools over the CP-Optimiser. The reasoning behind this is that OR-Tools is open-source and free to use, whereas IBM's solver is a commercial product. The availability of OR-Tools is why I think that OR-Tools is more suitable for scientific research.

3.2. Reschedule model evaluation metrics

To compare the different schedules with each other, two evaluation KPI metrics are defined based on several KPIs and some of the objectives mentioned in 1.3. This section focuses on evaluating a rescheduling model as a whole; the metrics that make up the objective value of the CP-model are defined in section 4.1.3 Which are defined as follows:

3.2.1. Quality evaluation metric

The quality metric is made up of several smaller metrics.

Turnaround time

The final KPI used in rescheduling is based on the turnaround time (TAT). The turnaround time refers to the duration an engine spends in the shop. Or more precisely, this is the number of days between the agreement of the work order and the moment an engine is signed off as finished and ready to return to the customer. Usually, when planners have to decide which engines to prioritise, they aim to minimise the maximum overdue TAT. However, in practice, this is often not the case due to other departments forcing other engines to be prioritised. These engines frequently need to be prioritised to improve the customer relationship, or if the associated overdue fee is too significant.

3.2.2. Robustness evaluation metric

The second metric corresponds to the robustness of a schedule, i.e., how resilient the schedule is to any other disruptions or deviations. The robustness is based on the **fever scores** and on how well a solution performs on a benchmark that checks the impact on a schedule after simulating disruptions.



Figure 3.1: Basic fever score function

Fever scores

The first metric to evaluate the robustness of a schedule is a metric based on the fever charts used **Critical Chain Project Management (CCPM)** [14]. Fever charts evaluate buffer consumption over the planned execution of the critical chain. I propose a grading function that determines the *hotness* of a resource for all shifts in a defined rolling window. The proposed method grades a resource at a point in time based on the remaining capacity. This not only indicates which resources are (almost) fully consumed at what points in time, but also translates this to a value in the same domain for every resource. This allows us to estimate what values the robustness metric will take, making it possible to scale other objectives accordingly.

For example, given the planned consumption $u_{r,t}$ and capacity $\mathcal{C}_{r,t}$ of a resource r at shift t with threshold X, the fever score $FS_{r,t}$ is defined by:

$$FS_{r,t} = \begin{cases} 0, & \rho_{r,t} < X, \\ FP_{\min}, & X \leq \rho_{r,t} < 1, \\ FP_{\max}, & \rho_{r,t} = 1, \\ FP_{\text{ex}} \cdot (u_{r,t} - C_{r,t}), & \rho_{r,t} > 1, \end{cases}$$

Meaning that the fever score of this example is as follows:

- No penalty is given if the consumption stays within a given bound.
- A fixed penalty FP_{min} is given if the consumption exceeds the given bounds, but the resource is not fully consumed.
- A fixed penalty FP_{max} is given if the consumption equals the capacity.
- A relative penalty is given by multiplying the excess consumption by FP_{ex}. Note that this penalty can only be applied to resources for which assigned consumption can exceed the capacity. FP_{ex} should be set a lot higher than FP_{max}, because the consumption should only exceed the capacity if there is no other way or if the quality would increase too significantly.

For these penalties, it holds:

$$0 < FP_{min} < FP_{max} < FP_{ex}$$

This example describes one of the simplest fever score functions; depending on the problem, scale, and complexity, one might want to adjust this function accordingly. The main idea is to divide the capacity into several sections and to penalise all resources at all points using values in the same domain (except when FP_{ex} is used to determine $FS_{r,t}$, in which case extremes can occur).

A lower fever score for a resource during a shift should indicate that the resource is less ”hot”, meaning that it is more likely that another job can be added/moved to this shift (or more resources from this job can potentially be moved to this shift)

As to why fever scores are only used with the corresponding rolling window, this was chosen for two reasons: the first reason was to mimic how short-term planning is currently made, which means that most jobs scheduled a week from now are more like drafts. Since disruptions occur frequently at any point in time, making exact plans too far in advance is of little use, as they are almost guaranteed to change due to disruptions.

The other reason is based on the hypothesis that by only determining the fever score for resources within the window, critical points will gather just outside the window, because within the window, they are more likely to be pushed later than earlier. This would increase the likelihood of similar jobs being

scheduled at comparable points in time, which should increase the likelihood of similar jobs exploiting the delay of the disrupted job to improve their own quality.

Disruption benchmark

The other robustness metric is based on a benchmark that takes the state of a schedule at a given shift, generates several sets of random disruptions of different sizes, applies the disruptions, and performs a reschedule. This provides information on how many jobs had to deviate by how much, as well as the change of the quality evaluation metrics, which corresponds to the robustness metrics **stability** and **quality robustness**, respectively. The final robustness metric, **repair costs**, is evaluated by comparing the search performance of the disrupted schedule and the non-disrupted schedule for the next step in time.

Another robustness method for measuring robustness is by benchmarking the results after disruptions. This benchmark starts by selecting various sets of jobs with similar start times in the reference input schedule.

Note that the disruption benchmark picks jobs that start in the reference data in the domain $[20, 50]$. *This is because the data at the start of the simulation is fixed, mainly because most jobs at the beginning are already in progress. However, the data at the end of the benchmark has a lot of spare room, as no new projects have been announced*.

If these jobs occur close to each other in the reference schedule, they will likely be close to each other in the found solutions as well. Then, for each job in each set of jobs, a random disruption is assigned to that job, and these same sets of disruptions are then applied to each instance being benchmarked. The benchmark begins two shifts before the planned start of the first job with a disruption, and it ends five shifts after the final disruption has occurred.

Then the impact of the disruptions can be evaluated based on:

- **(Average) change in any metric from start to end.** Provides insight into how well a solution was able to sustain the applied disruption(s).
- **(Average) change in any metric relative to non-disrupted solutions.** Potentially better insight into how well a solution was able to sustain the applied disruptions by using the change in value of the corresponding metric of the non-disrupted schedules as a reference to how much the value of the corresponding metric would change if there were no disruption(s) introduced.

4

Proposed model

In this chapter, the model will be defined, as well as everything that makes up the proposed rescheduling model.

4.1. Model definition

The proposed model is used for both optimising schedules and estimating a valid reference schedule. In order to find optimal schedules, weights have to be assigned to the three sub-objectives. In general, the order of importance for the different sub-objectives is **quality** $\rightarrow$ **deviation** $\rightarrow$ **robustness**. However, when estimating valid reference schedules, the robustness objective is ignored and the main focus is put on the deviation metric (deviation $\rightarrow$ quality). This way the model will attempt to meet the reference schedule, and when conflicts are encountered, it aims to resolve these using minimal changes without spreading every task over the time horizon.

4.1.1. Notation

Sets and Indices

Symbol	Definition
$\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_N\}$	Set containing all N projects.
$t \in \mathcal{T} = \{0, 1, \ldots, H\}$	Set of all shifts in the entire planning horizon of length H.
$t_0 \in \mathcal{T}$	Starting shift used for the rolling time horizon.
$\mathcal{J}_p$	Set of tasks in project p, indexed by j.
$\mathcal{R} = \mathcal{R}^W \cup \mathcal{R}^F$	Set of resources, indexed by r.
$\mathcal{R}^W$	Subset of workstation resources ($\mathcal{R}^W \subset \mathcal{R}$).
$\mathcal{R}^F$	Subset of workforce resources ($\mathcal{R}^F \subset \mathcal{R}$).
$\mathcal{R}^W \cap \mathcal{R}^F = \varnothing$	Workforce and workstation sets are disjoint.

Model Parameters

Symbol	Definition
$T_{\max}$	Length of planning horizon defined as number of shifts.
$\mathcal{P}_j^{\text{start}}$	Earliest shift a task j of project p can begin.
Π_p	Profile associated with project p.
τ_p^e	Engine type of project p.
τ_p^α	Project type (assembly/disassembly).
τ_p^w	Workscope type (major/minor/etc.).

Task Parameters

Symbol	Definition
$r_{p,j}^{start}$	Last assigned starting shift for task j of project p.
$r_{p,j}^{end}$	Last assigned ending shift for task j of project p.
$r_{p,j}^{dur}$	Defined duration for task j of project p.
$\tau_{p,j}$	Type of task j in project p.
$\delta_{\text{cost}}(\cdot)$	Deviation cost function.

Symbol	Definition
L^{min}	Minimal time lag.
L^{max}	Maximal time lag.
$FS_{j,k}^L \in D^{FS}$	Finish-start dependency with time lags for project j and k.
$SS_{j,k}^L \in D^{SS}$	Start-start dependency with time lags for project j and k.
$FF_{j,k}^L \in D^{FF}$	Finish-finish dependency with time lags for project j and k.
$SF_{j,k}^L \in D^{SF}$	Start-Finish dependency with time lags for project j and k.
$D = D^{FS} \cup D^{SS} \cup D^{FF} \cup D^{SF}$	Set of all dependencies in the environment

Ordering Dependency Parameters

Resource Parameters

$\mathcal{C}_{r,t}^{\text{doc}}$ Documented capacity of resource r at shift t.

$R(p,j,r) = \{\rho_{p,j,r,0}, \rho_{p,j,r,1}, \ldots, \rho_{p,j,r,\theta}\}$ Defined requirement of a resource r for task (p,j) over the defined duration.

$\rho_{p,j,r,\theta}$ Defined requirement of a resource r for task (p,j) during its θ-th shift of execution.

$\rho_{p,j,r,\theta}^{lb}$ Defined lower bound of the scheduled consumption of a resource r for task (p,j) during its θ-th shift of execution.

$\rho_{p,j,r,\theta}^{ub}$ Defined upper bound of the scheduled consumption of a resource r for task (p,j) during its θ-th shift of execution.

Decision Variables

$\mathcal{S}_{p,j}^{start} \in \mathbb{Z}_{\geq 0}$ Planned start shift of job j in project p.

$\mathcal{S}_{p,j}^{end} \in \mathbb{Z}_{\geq 0}$ Planned end shift of job j in project p.

$\mathcal{S}_{p,j}^{dur} \in \mathbb{Z}_{\geq 0}$ Planned duration in shifts for job j in project p.

$x_{p,j,t} \in \{0,1\}$ Equals 1 if job (p,j) is active in shift t, 0 otherwise.

$\forall u_{p,j,r,t} \in \mathbb{Z}_{\geq 0}$ Planned consumption of resource r in shift t by job j of project p

4.1.2. Constraints

Project release times: For each project, the starting shift cannot be earlier than $\mathcal{P}_j^{\text{start}}$

$$\mathcal{P}_j^{\text{start}} \leq min(\mathcal{S}_{p,j}^{start}) \quad \forall p \in \mathcal{P},\, j \in \mathcal{J}_p \tag{4.1}$$

Task duration: For each task, the duration should equal end - start

$$\mathcal{S}_{p,j}^{dur} = \mathcal{S}_{p,j}^{end} - \mathcal{S}_{p,j}^{start} \quad \forall p \in \mathcal{P},\, j \in \mathcal{J}_p \tag{4.2}$$

Dependency constraints: For all temporal dependencies $d_{j,k} \in D$ with time lag l in project p.

$$d_{j,k} \in D^{FS} \quad \mathcal{S}_{p,j}^{end} + l^{min} \leq \mathcal{S}_{p,k}^{start} \leq \mathcal{S}_{p,j}^{end} + l^{max} \qquad \forall (j,k) \tag{4.3}$$

$$d_{j,k} \in D^{SS} \quad \mathcal{S}_{p,j}^{start} + l^{min} \leq \mathcal{S}_{p,k}^{start} \leq \mathcal{S}_{p,j}^{start} + l^{max} \qquad \forall (j,k) \tag{4.4}$$

$$d_{j,k} \in D^{FF} \quad \mathcal{S}_{p,j}^{end} + l^{min} \leq \mathcal{S}_{p,k}^{end} \leq \mathcal{S}_{p,j}^{end} + l^{max} \qquad \forall (j,k) \tag{4.5}$$

$$d_{j,k} \in D^{SF} \quad \mathcal{S}_{p,j}^{start} + l^{min} \leq \mathcal{S}_{p,k}^{end} \leq \mathcal{S}_{p,j}^{start} + l^{max} \qquad \forall (j,k) \tag{4.6}$$

$$\tag{4.7}$$

Resource consumptions For each planned consumption $u_{p,j,r,t}$ the following hold:

$$x_{p,j,t} = 0 \;\Rightarrow\; u_{p,j,r,t} = 0 \qquad \forall p,j,r,t \tag{4.8}$$

$$x_{p,j,t} = 1 \;\Rightarrow\; \left(u_{p,j,r,t} = 0 \;\vee\; \rho_{p,j,r,\theta}^{lb} \leq u_{p,j,r,t} \leq \rho_{p,j,r,\theta}^{ub}\right) \qquad \forall p,j,r,t \text{ with } \theta = t - \mathcal{S}_{p,j}^{start}. \tag{4.9}$$

Resource capacities. Let $T_H = \{t_0, \ldots, t_0 + H - 1\}$ be the current rolling horizon. For each activity (p,j), resource r, and shift $t \in T_H$:

- $x_{p,j,t} \in \{0,1\}$ indicates whether job (p,j) is executing at time t (as linked to its start time and duration in §**??**).
- $a_{p,j,r} \in \mathbb{Z}_{\geq 0}$ is the per-period demand of resource r when (p,j) is active.

We also define the instantaneous utilisation

$$u_{r,t} \;=\; \sum_{p \in P} \sum_{j \in J_p} a_{p,j,r}\, x_{p,j,t} \qquad (r \in R,\; t \in T_H).$$

We distinguish between hard workstation resources R_W and workforce resources R_F, which we allow to be exceeded in a soft manner. However, exceeding the available capacity is neither desired nor realistic to do constantly, which is why the fever score definition from Section 3.2.2 increases the penalty for exceeding capacity based on the significant shortage of the resource. The reasoning for allowing the workforce to exceed the workforce capacities while limiting workstations is that the workforce is something flexible. The mechanics on the floor regularly help each other out if they have leftover time and hands. If we wanted to allow something similar for workstations, it would somehow have to be possible to make additional workstations appear.

Additionally, unlike most RCMPSP/max problems solved using CP, the cumulative constraint is not employed. This has two underlying reasons, the first being that this model focuses on rescheduling with a rolling horizon. By using a time-indexed approach, hints can be easily set and applied based on the last known schedule. The other reason is that the defined resources do not always have the same capacity at each point in time.

Hard-capacity resources (enforced workstation limits) ($r \in R_W$).

$$\sum_{p \in P} \sum_{j \in J_p} a_{p,j,r}\, x_{p,j,t} \;\leq\; C_{r,t} \qquad (\forall r \in R_W,\; \forall t \in T_H). \tag{4.10}$$

Soft-capacity workforce ($r \in R_F$) (penalized workforce limits). We allow bounded over-assignment via nonnegative slack $s_{r,t}$ that is penalised by the robustness component Z_r:

$$\sum_{p \in P} \sum_{j \in J_p} a_{p,j,r}\, x_{p,j,t} \;\leq\; C_{r,t} + s_{r,t} \qquad (\forall r \in R_F,\ \forall t \in T_H), \tag{4.11}$$

$$s_{r,t} \;\geq\; 0. \tag{4.12}$$

4.1.3. Objective Function

Since the objective of the model is not just to find a schedule with the highest quality, it has to find the highest quality robust reschedule. Meaning that the objective consists of three parts: **Quality**, **Deviation** and **Robustness**.

Quality The quality objective is made up of the quality value of each project. Depending on the stage of the project and the difference between contract TAT and actual TAT, these projects are evaluated.

If a project is currently busy with assembly, the quality of the project corresponds to how many shifts the actual TAT exceeds the contract TAT. This prioritises projects which have the most delay.

If the number of shifts the actual TAT exceeds the contract TAT exceeds some constant, it is marked as too disruptive to evaluate on that value. In these and all other scenarios, is the project evaluated on minimising the make span?

Deviation The deviation objective is used to penalise the changes of all jobs in the schedule. The deviation penalty depends on two factors. The first is the type of deviation. A job can change duration and/or change the start and end timings. When the duration of a job changes, the deviation is penalised in every instance of the simulations. Changing job durations is not favourable, so if it can still be avoided later on, that should be done. When the start and end times change, the penalty only occurs once (when the change takes effect). For both of these types of deviations, there is also a scale for the penalty based on the distance in the time horizon. Making adjustments in the near future is less favourable and should be avoided, while changes occurring further in the time horizon can usually be made easily.

Robustness The robustness metric is used to avoid exceeding resource capacities and allow for more robustness in the schedule. The metric scores each resource at each shift on the time horizon based on the consumption relative to the capacity. An in-depth explanation can be found in section 3.2.2.

Multi-objective optimisation in CP models is usually achieved by using either weighted sum or by hierarchical optimisation. The weighted sum method assigns a weight to each objective to construct a combined Pareto front. Whereas hierarchical finds the optimal solution for the first objective and then reruns the model for another objective, bounding the previous objective based on the derived corresponding objective value. This causes it to create a new Pareto front for each objective used.

Performing hierarchical optimisation requires an order of objectives and threshold definitions for each. The order contains the objectives that are solved, along with the order in which the models are solved. The threshold defines the bound for the objective in the successive iterations. Based on these requirements, it was assumed that hierarchical optimisation would be less suitable for rescheduling than weighted sum. Since for rescheduling, the optimisation would start with either *quality* or *deviation* (since solving for just robustness would just spread out all jobs over the time horizon). If the first objective were quality, the model would seek the highest quality solution, without taking into account how much the solution differs from the solution at a previous point in time. If the first objective were deviation, it would likely focus on copying the previous solution too much. And for every instance where it is able to copy the exact solution, the deviation objective would become 0, which would likely prevent any deviations at the next step of the optimisation.

This is why the weighted sum was chosen as a multi-objective optimisation method.

Symbol	Description
Z	Objective function
Z^q	Quality objective function
Z^d	Deviation component of the objective
Z^r	Robustness component of the objective
w_q	Integer weight associated with Z^q
w_d	Integer weight associated with Z^d
w_r	Integer weight associated with Z^r
w_{qr}	Additional weight based on the number of items that make up Z^r, more details in section 5.1.3

Table 4.1: Objective function variables

$$\min Z \quad = \quad w_q * w_{qr} * Z^q + w_d * Z^d + w_r * Z^r \tag{4.13}$$

$$\text{s.t.} \quad w_q, w_d, w_r \in \mathbb{Z}_{\geq 0} \tag{4.14}$$

4.1.4. Objectives

The model objective is made up of several sub-objectives: *Schedule quality*, *Schedule deviation* and *Schedule robustness*

Schedule quality focuses on how profitable the schedule is on paper. This objective is made up of:

1. Difference in turnaround time
2. Executed workload

Schedule deviation is included to control changes to the schedule. By deriving a *deviation penalty* for each deviation of job of each project based on:

1. Magnitude of deviation (how large is the deviation, which aspects have deviated)
2. Distance on the time horizon (prevent or heavily penalise deviations that happen in the near future, little to no penalty for deviations that happen near the time horizon)

These penalties are then used both summed up, as well as the maximum values of all penalties. The max value provides insight into the largest deviation, and combined with the summed penalties gives an insight to how much the schedule deviated overall.

Finally, *Schedule robustness* focuses on handling the disruptive nature of the context. By focusing on finding a schedule that is able to handle any kind of deviation the best way it can.

4.2. Model description

4.2.1. Experimental setup

The model defined in 4.1.3 was implemented and executed using ORTools and Python 3.10.

The experiments were executed on a system with:

- Windows 11 Pro
- 32GB RAM
- AMD Ryzen 7 3700X 8-Core @ 3600Mhz

4.2.2. Implementation

The model was implemented using several components. The first component processes each available snapshot of the live schedule from Anago by:

1. Retrieve all jobs and events from the snapshot.

2. Using reference profiles, expand the parsed snapshot by extending jobs that have been cut off by the snapshot and adding jobs that will happen outside the snapshot in the future.
3. Retrieve values for the resource variables (employees and workstation).

Once all snapshots of the schedules have been processed, they are combined into a single object, which is used as reference data for the remainder of the model.

Before the reference data is passed on the next component, is some data validation and repair performed. This is done both manually and automatically. The automatic validation focuses on whether the projects correspond to the provided reference profiles, and if not, determine whether manual input is required or if the corresponding profile can be adjusted to meet the given data. Manual input is only required when projects contain nonsense jobs which do not align with the type of project. These jobs are usually accidents, for instance, an engine assembly project that contains a module disassembly job is usually just entered incorrectly. Automatic repairs focus mainly on the dependencies between jobs. For example, a profile might indicate that a task B can only start once task A is completely finished. But in the input data starts task B two shifts before task A has ended. In these scenarios, the schedulers explicitly allowed this, which is why the assumption was made that for this project the dependency between these job can be changed to correspond to the input schedules. Once the input data has been validated and repaired, is it mapped to a format that matches the model definition from 4.1.3 and is used as input for the next component.

4.2.3. Simulation

For each schedule in each experiment, a simulation was done to compare the different schedules under different circumstances at different moments in time. This simulation can be defined as follows:

Symbol	Definition
H	Length of simulation horizon defined by number of shifts.
S_t^i	State of the schedule for instance i at when observed at shift t
D_t	All disruptions of the schedule known at D_t
$f(S_t^i, D_t) = S_t^{i+t}$	Transformation f of the schedule at shift t with the introduction of disruptions in D_t
$Z_{i,t}^q$	Objective value for the quality metric for schedule i when observed at shift t
$Z_{i,t}^d$	Objective value for the deviation metric for schedule i when observed at shift t
$Z_{i,t}^r$	Objective value for the robustness metric for schedule i when observed at shift t

5

Experimental evaluation

5.1. Environment modelling and schedule evaluation

Since the basis of this thesis is a CP model that can find schedules of high quality, with minimal short-term deviations, and aims for high robustness to handle any unexpected disruptions. This initial model had to be designed, implemented and optimised to run within designated time bounds. Once this was done, an additional feature was added to the model that allowed jobs to deviate in duration as well as the assigned workforce at different shifts.

To compare the schedules generated by the optimal model, a reference model had to be defined as well. This was necessary because the original reference schedules often contained conflicts, which would be solved later in the actual MRO. This reference model uses the same model as the optimal model, but with a different objective, lacks robustness, and does not include any extra features that the optimised model adds. By setting the objective of the reference model to focus on minimising the deviation, followed by minimising the quality, the resulting reference solutions are as close to the actual schedules as possible.

The goal is to demonstrate that, even as the problem becomes more complex by considering robustness and allowing jobs to vary both in duration and resource allocation, through rescheduling, the model is still able to find good solutions within a reasonable time.

5.1.1. Optimised and Reference model descriptions

As mentioned, the optimal and reference models use the same basis. The key difference is that the optimised model takes quality, deviation, and robustness into account, allowing jobs to have dynamic durations and enabling dynamic resource allocation. Whereas the reference model only considers quality and deviation, and does not allow for dynamic durations or dynamic resource allocation.

This resulted in the following requirements for the parameters of both models.

- For reference models, w_d should always be higher than w_q. The focus is on finding a feasible schedule as similar to the reference data as possible.
- For optimised models, w_d should always be at most w_q. Because the optimised model should focus more on how higher solution quality can be achieved by deviating from the previous solution.

Note that for the model parameter evaluation, several instances were solved that do not meet these requirements. This is because those instances were then used to verify the correctness of the requirements.

5.1.2. Method

Model parameters

For the first stage, several parameters had to be determined and set. For the solver, an appropriate timeout had to be chosen such that it would find near-optimal solutions nearly all the time. For the model,

a basis for assigning weights to the sub-objective values had to be derived. Both of these parameters were derived by running multiple instances of the model for an entire simulation, while keeping track of each intermediate solution found, including the time elapsed, the input used, and the resulting objective values.

For the initial parameter exploration, the timeout was set to 90 seconds for every parameter configuration. This was because we had to solve 71 instances for each configuration, and the goal at this point was to identify the characteristics and bottlenecks caused by which combinations of parameters.

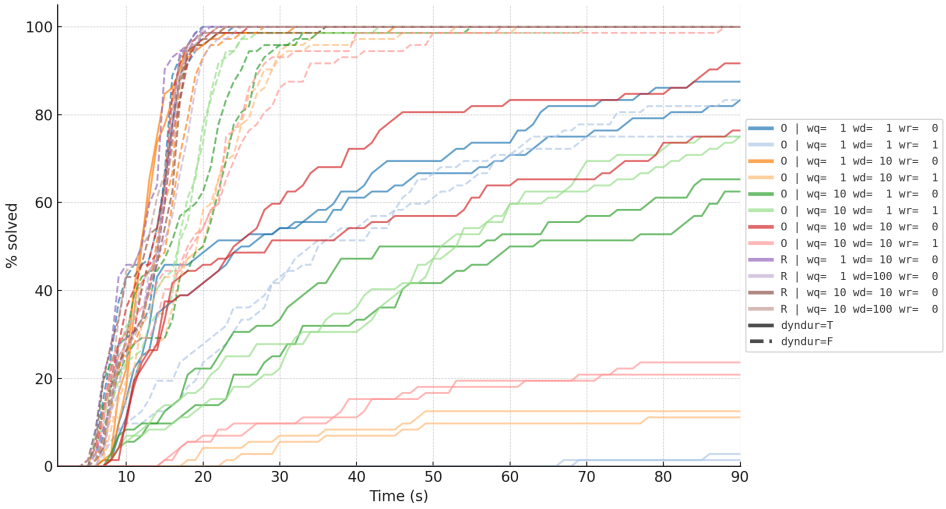


Figure 5.1: Percentage of instances solved over time for various parameter configurations

The first insight that can be seen in figure 5.1, is that using dynamic job durations (dashed lines) has a significant impact on the time needed to find and show optimality.

Something else that stands out is the time needed for instances with $(w_q, w_d, w_r) = (1, 1, 1)$, which is significantly worse when compared to any of the other configurations. Upon closer inspection of the intermediate results and the individual values for (Z_q, Z_d, Z_r). Here, it shows that the values of Z_r are of similar magnitude to Z_q, which would explain the extended search duration. For instance, the solver finds the first feasible solution with $Z_q = Z_r$. Then it is likely easier to decrease the value of Z_r, as this can be achieved by simply moving some jobs to a point later in the future. However, moving those jobs to a later point in time would increase not only Z_q, but also Z_d. And upon inspecting the various schedules made for the instances with $(Z_q, Z_r) = (1, 1)$ can be seen that the projects are indeed spread out over the horizon. For this reason,

Based on this information, a timeout was set to 120 seconds, and $w_q = 10$, $w_d = 10$ and $w_r = 10$ for the default, where there is no strong preference for one of the objectives, but all are more or less the same.

Improving the model

The primary property of the environment that was taken into account when seeking ways to improve the model's runtime (and thus the quality of solutions found within the time limit) was that the model had to perform rescheduling.

Value hinting Since the model would always have access to how the schedule looked in previous moments in time, it can start the search by setting the variables to the value of that schedule. This shows that adding value hints to the variables significantly improved the search, requiring

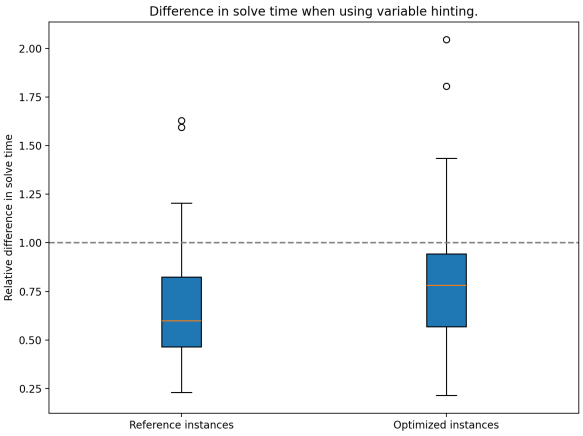


Figure 5.2: Speed-up of solve times when using value hints

experiment type	w_q	w_d	w_r	joined	dynamic duration	use fever	% solved by time (s)								
							10	20	30	40	50	60	70	80	90
O	1	1	0	F	F	F	29.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	1	1	0	F	T	F	11.1	41.7	54.2	62.5	69.4	73.6	81.9	86.1	87.5
O	1	1	0	T	F	F	43.1	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	1	1	0	T	T	F	15.3	48.6	54.2	61.1	66.7	70.8	75.0	79.2	83.3
O	1	1	1	F	F	T	9.7	23.6	43.1	51.4	66.7	70.8	75.0	75.0	75.0
O	1	1	1	F	T	T	0.0	0.0	0.0	0.0	0.0	0.0	1.4	1.4	1.4
O	1	1	1	T	F	T	9.7	27.8	41.7	54.2	61.1	70.8	77.8	81.9	83.3
O	1	1	1	T	T	T	0.0	0.0	0.0	0.0	0.0	0.0	1.4	1.4	2.8
O	1	10	0	F	F	F	26.4	95.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	1	10	0	F	T	F	18.1	95.8	98.6	100.0	100.0	100.0	100.0	100.0	100.0
O	1	10	0	T	F	F	29.2	93.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	1	10	0	T	T	F	22.2	95.8	98.6	98.6	100.0	100.0	100.0	100.0	100.0
O	1	10	1	F	F	T	15.3	58.3	93.1	97.2	98.6	98.6	100.0	100.0	100.0
O	1	10	1	F	T	T	0.0	0.0	5.6	6.9	9.7	9.7	9.7	11.1	11.1
O	1	10	1	T	F	T	19.4	50.0	94.4	98.6	98.6	100.0	100.0	100.0	100.0
O	1	10	1	T	T	T	0.0	4.2	6.9	8.3	12.5	12.5	12.5	12.5	12.5
O	10	1	0	F	F	F	25.0	50.0	95.8	98.6	98.6	100.0	100.0	100.0	100.0
O	10	1	0	F	T	F	5.6	13.9	25.0	33.3	41.7	50.0	51.4	52.8	62.5
O	10	1	0	T	F	F	29.2	62.5	95.8	100.0	100.0	100.0	100.0	100.0	100.0
O	10	1	0	T	T	F	8.3	22.2	33.3	47.2	50.0	52.8	56.9	61.1	65.3
O	10	1	1	F	F	T	23.6	77.8	98.6	98.6	98.6	98.6	100.0	100.0	100.0
O	10	1	1	F	T	T	6.9	13.9	22.2	30.6	47.2	59.7	69.4	72.2	75.0
O	10	1	1	T	F	T	31.9	73.6	98.6	98.6	100.0	100.0	100.0	100.0	100.0
O	10	1	1	T	T	T	5.6	18.1	27.8	36.1	47.2	59.7	62.5	68.1	75.0
O	10	10	0	F	F	F	30.6	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	10	10	0	F	T	F	9.7	41.7	59.7	72.2	80.6	83.3	83.3	84.7	91.7
O	10	10	0	T	F	F	36.1	97.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0
O	10	10	0	T	T	F	11.1	45.8	51.4	54.2	56.9	63.9	65.3	73.6	76.4
O	10	10	1	F	F	T	27.8	54.2	86.1	93.1	98.6	98.6	98.6	98.6	100.0
O	10	10	1	F	T	T	0.0	6.9	9.7	11.1	16.7	19.4	20.8	23.6	23.6
O	10	10	1	T	F	T	30.6	54.2	93.1	98.6	98.6	100.0	100.0	100.0	100.0
O	10	10	1	T	T	T	0.0	5.6	9.7	15.3	18.1	18.1	19.4	20.8	20.8
R	1	10	0	F	F	F	29.2	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	1	10	0	T	F	F	45.8	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	1	100	0	F	F	F	25.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	1	100	0	T	F	F	30.6	93.1	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	10	10	0	F	F	F	26.4	95.8	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	10	10	0	T	F	F	43.1	97.2	98.6	100.0	100.0	100.0	100.0	100.0	100.0
R	10	100	0	F	F	F	27.8	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0
R	10	100	0	T	F	F	44.4	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 5.1: Solve-time coverage at 10s steps up to 90s.

less time and fewer intermediate solutions to find optimal values.

Rescheduling instance	Mean	Median	Std	Min	25%	75%	Max
Reference solution	0.671	0.599	0.306	0.230	0.465	0.824	1.629
Optimized solution	0.809	0.782	0.346	0.215	0.569	0.943	2.046

Table 5.2: Relative time needed to solve for optimality with hints and without hints

As can be seen in table 5.2, does type hinting improve solve time significantly in most cases, with almost 33% speed-up for reference solutions and 19.1% for optimised solutions.

Deviation weights As can be seen in 5.1, there is not a single instance finding optimality quickly when setting w_d to a low value. This does make sense, since the weighted objective value $w_d * Z^d$ is a direct bound on the search space. Since this value scales based on w_d and the size of deviations at each iteration. This does pose a challenge for selecting appropriate weights for the optimised model. If w_d is set too high, the model starts to mimic

w_q	w_d	w_r	Fever	Joined	Dynamic	μ-eq	μ-net-avg
1	1	0	F	F	F	49.2500	-0.0370
10	10	0	F	F	F	49.2500	-0.0370
10	1	1	T	F	T	49.2222	-0.0278
1	10	1	T	F	F	49.2639	-0.0972
1	10	0	F	F	F	49.2639	-0.0972
10	10	1	T	T	T	48.6944	0.2344
10	1	1	T	T	T	49.1250	-0.1620
1	10	0	F	T	F	49.2222	-0.1898
10	10	0	F	T	F	49.2222	-0.1898
1	10	1	T	T	F	49.2222	-0.1898

Table 5.3: Relation between the average number of projects that retained their previously planned time

the reference data, whereas if it is set too low, the search space explodes and the solver will ignore scheduling towards a schedule similar to the previous schedule.

5.1.3. Insights from solving

Besides a comparison of statistics related to the solutions, the intermediate solutions were also investigated to look for possible exploitations. The most important insights found are:

1. Reason for slower solve times with worse objective values at the first few iterations of the simulation
2. What parts of the model perform better thanks to the variable value hints.

The first insight is that the solver needs to solve several iterations first to properly make the schedule feasible again, without deriving too much from it. As described in 3.1, the provided reference schedules are infeasible due to impossible resource consumptions. The goal of the simulation is to optimise the quality of the solution while keeping the deviation from the previous one to a minimum. However, when the simulation starts, a relatively high number of jobs have to deviate to make the solution a feasible schedule. This causes the simulation to deviate, thereby improving the solution quality. This is why each simulation has an extended timeout for the first three iterations, which helps in finding a better starting point.

A similar insight had to do with the balance $\frac{w_q}{w_d}$. If the quality objective is weighed too heavily relative to the deviation metric, it tends to overtake the other objectives. This leads to large and frequent deviations, since the larger $\frac{w_q}{w_d}$ is, the more a solution will deviate for a better quality solution. On top of this, since the optimised model also includes the robustness metric, if $\frac{w_q}{w_d}$ becomes too small, the robustness metric will overtake as priority, which causes large and frequent deviations without consistently improving the quality. To weaken this effect, w_{qr} is introduced as an additional weight for the quality objective. The weight w_{qr} is based on the number of shifts in the rolling fever window for each resource.

Another insight is about how using hinting helps with directing the solver to perform a near-optimal rescheduling quickly. When examining the intermediate solutions, it becomes apparent that once the solver has found a feasible solution, it can soon find a nearly optimal solution within seconds. This makes sense, since if, after hinting at the timing of each job, a conflict results, then the conflict can likely be traced back to a set of conflicting jobs. Additionally, the next chapter discusses the impact of incorporating robustness into the model on the solution quality.

5.1.4. Performance evaluation

To conclude this part, a brief performance evaluation indicates that once projects are scheduled, they rarely deviate from their assigned tasks. Making variable hints the most critical performance improvement method for our problem



Figure 5.3: Simple visualisation for deciding fever scores. If the value is green, assign no fever. Assign a small fever score to values in yellow, A significant penalty for exceeding the capacity. And a ramping penalty for entering red

5.2. Robustness objectives in CP modelling

As shown in section 2.3.1, robustness metrics can be interpreted and measured in various ways. However, in most cases, implementing these exact values in a CP model would negatively impact the runtime, especially for more complex problems, such as MRCMPSP. This is the reason why alternative additions are implemented.

5.2.1. Robustness metrics

For the first robustness metric, *Stability*, the focus lies on minimising the dependency of projects on other projects. If a job is disrupted and has to deviate, likely, other jobs in the same project that depend on it will also deviate. But that does not mean that other projects should suffer from this deviation as well.

For the second robustness metric, *Quality robustness*, the easiest method would be to aim for buffer capacities that are sufficient for all jobs. This would allow jobs to deviate, while retaining the original objective value. However, this would also mean that the original objective value is worse than it would be if the buffers were not considered. Another aspect that must be taken into account is that the quality loss of outgoing projects is more severe than that of incoming projects. If the disassembly process is somewhat delayed, Gate 3 may be able to compensate for this delay during parts repair. Delays in the assembly process directly impact the project turnaround time. However, if assembly were always prioritised over disassembly, then eventually there might not be enough work entering gate 3 to utilise the available capacity and resources fully.

For the final metric, *Flexibility*, a solution is already proposed in 4.1.3. As shown in table 5.2, the time needed for a reschedule heavily depends on whether the hinted values provide correct starting points of the search. Thus, by focusing on stability, the number of jobs able to retain their initial schedule after a disruption would be maximised, and hence, more hinted values would correspond to their optimal values.

This is why the focus for this robustness metric is on finding both an appropriate robustness objective value weight calculation and extending the model in such a way that it aims for a solution that balances deviations in disassembly and assembly while prioritising the outgoing flow from assembly processes.

To add robustness to the schedule, I propose incorporating a metric based on fever charts (source: critical chain). By introducing a penalty based on how much relative capacity is left of each resource in the following 20 shifts (two weeks). After testing around a bit, I decided to set the aim for available resources during each change in the upcoming time window to be 20%. Then, based on the consumption, a penalty is added.

To evaluate whether the optimised solutions perform better than the reference solutions, simulations are performed as described in chapter 5.3.2.

5.2.2. Impact of robustness constraints

To determine the impact of the robustness constraints, a simulation is performed using three instances with identical parameters. Each instance does have its own weights for the objective value. The first instance (quality-focused) aims to perform rescheduling with little to no attention to robustness (essentially running the simulation without it), using $w_q = 80, w_d = 4, w_r = 1$ as weights. The other two instances (robustness-focused) are relatively similar and have comparable performance, using weights of $w_q = 4, w_d = 2, w_r = 1$ and $w_q = 3, w_d = 2, w_r = 1$, respectively. The comparison between the results of the different models can been found in tables 5.4 and 5.5.

Table 5.4 compares the statistics between the quality-focused solutions and the solutions with robust-

Metric	Robust A ($w_q = 3, w_d = 2, w_r = 1$) vs Quality	Robust B ($w_q = 4, w_d = 2, w_r = 1$ vs Quality
Total number of projects compared	50.00	50.00
Number of projects robust finished earlier	16.00	7.00
Number of projects quality finished earlier	26.00	37.00
Number of projects with same end time	8.00	6.00
Δ end (Robust - Quality): mean	2.18	7.76
Δ end (Robust - Quality): P90	11.00	18.10
Σ Δ end (Robust - Quality): sum	109.00	388.00

Table 5.4: Per-project completion comparison: optimised models with a focus on robustness vs an optimised model $w_q = 80, w_d = 40, w_r = 1$ that only uses robustness to break ties. Negative Δ indicates finetuned finishes earlier. Includes the sum of end-time deltas.

Experiment	Number of data points	Mean	Median	Std. dev.	P90	P95
Robust A ($w_q = 3, w_d = 2, w_r = 1$)	21600	1.83	0.00	22.98	3.00	10.00
Robust B ($w_q = 4, w_d = 2, w_r = 1$)	21600	2.38	0.00	46.41	3.00	10.00
Quality ($w_q = 80, w_d = 40, w_r = 1$)	21600	**41.89**	0.00	215.28	10.00	120.00

Table 5.5: Fever score statistics per experiment (lower is better). Includes P90 and P95.

ness. As can be seen, the quality-focused solution can typically complete projects at an earlier shift than the robust solutions. Especially, robust solution B. Whereas A most likely has a few outliers as indicated by the 90th percentile. Table 5.5 shows statistics on the fever scores the different models received for their solutions. This clearly demonstrates that taking robustness into account helps reduce the critical point in time for resources. Whereas B performs slightly worse than A (likely due to some unfortunate decision, based on that B has the same standard deviation, median, 90th percentile and 95th percentile, while still having a 30% higher mean.) do both significantly outperform the quality focus solutions.

5.3. Evaluation under disruptions

After implementing the model described in chapter 5.2 and chapter 5.2, a validation is needed to compare the reference schedule with the schedules optimised to handle disruptions, to achieve this, a simulation is proposed that utilises mock disruptions to define various disruption scenarios, which are then performed and evaluated.

5.3.1. Simulating disruption scenarios

The following experiment is proposed for the evaluation. First, for each model to be used in finding schedules, the parameters must be set. Starting with the model M_{REF} that mimics the reference schedule. These instances require *dynamic durations and dynamic resource consumptions* to be disabled. For the objective weights, any setting that has w_d set to a value that significantly outweighs the other instances and where $w_r \in \{0, 1\}$. After trying various settings, the weights were set to $(w_q = 10, w_d = 100, w_r = 0)$.

For the optimised instances, we reuse the same configurations as we used in 5.2 to define the models M_A and M_B. Using objective weights $(w_q = 3, w_d = 2, w_r = 1)$ and $(w_q = 4, w_d = 2, w_r = 1)$ respectively. Both instances have *dynamic durations and dynamic resource consumption* enabled. This is because these configurations performed well in previous experiments, and the resulting schedules have been frequently inspected and validated.

Then, for each of the three models, a complete simulation is performed without any disruptions, saving the best-found solution at each step of the simulation to a file. These saved solutions serve as a baseline for simulating disruptions, which is why it was essential to ensure that a high-quality solution was stored for each step of the simulation. This was ensured by first changing the time-out from two minutes to four minutes, or after no new solutions are found within the last 90 seconds, and at least two minutes have passed. To ensure accuracy, each saved solution was reloaded and solved again from that point in time to verify that no significantly higher-quality solutions were found.

Next, mock disruptions had to be generated. The initial plan was to have four types of disruptions. **Job-**

start disruptions, **Job-duration disruptions**, **Workstation disruptions** and **Workforce disruptions**.

Job-start disruptions mimic disruptions like delivery delays and engines not arriving at the MRO on time. To simulate these events, the start and end times of the disrupted job were adjusted accordingly. These disruptions can only occur for jobs that have not yet started at the moment the disruption takes place.

Job-duration disruptions mimic disruptions that occur when jobs encounter problems and take longer than initially planned, or when jobs, such as engine assembly, have already started and are waiting for a delayed module. Job-duration disruptions only occur in jobs that have already started, this way the beginning of the job is already locked forcing the disrupted job to stay at the scheduled moment (like an engine that is already being assembled in an engine stand).

The other two disruptions were implemented, but were eventually removed from the simulation. Workstation disruptions were the first to go; these would scale down the available capacity for a given period of time based on a randomly selected scalar value. These disruptions would often result in infeasible solutions, because workstation consumptions cannot exceed their defined capacity. And if the goal is to reduce the available number of a specific type of workstation, then this could also be achieved in a safer way, for example, by introducing job-duration disruptions.

The final disruption type was the workforce disruption. These disruptions were designed to simulate events such as employees calling in sick. Mainly disruptions that happen somewhere between now and the following four shifts. These disruptions would rarely affect the schedule at all. In most cases, the affected jobs had already started. This not only limits what can be changed about the schedule but also any deviations that do occur and require other jobs to adjust accordingly. These deviations occur in the most heavily penalised window, which means that the cost of rescheduling would easily outweigh the cost of assigning too much work to the available workforce.

To simulate under disruptions, a definition of the disruption densities is established. This definition describes the number of scenarios that should be simulated, where the corresponding number of disruptions occur in close proximity to each other. For example, $2 \rightarrow 3$ means that three scenarios are performed with two disruptions happening near each other. To do so, for every scenario, a random point in time $T \in [20, 21, \ldots, 49, 50]$ is selected. This window is used to avoid the bounds of the reference input schedule. Then the required number of jobs to disrupt is chosen from the jobs that start at T. If there are not enough jobs starting at T, select the remaining jobs from $T+1, T+2, \ldots$. After defining the densities, the number of repetitions is defined, which indicates how often these benchmarks are performed (each repetition uses a different random seed to select T and corresponding jobs).

Finally, a simulation was run for each scenario on each model. The window of this simulation was set to two shifts before the planned start of the first disrupted job, and the window end was put to two shifts after the start of the last disrupted job (which means that the windows can differ for each of the models, since each model scheduled the jobs at a different point in time)

Once all disruption scenarios have finished, the different models are evaluated on how well the disruptions were handled.

5.3.2. Robustness evaluation

Various aspects of the solutions must be compared to determine how robust the solutions found for a model are.

The evaluation focuses on the following aspects: **resource management**, **quality robustness** and **schedule stability**. Resource management evaluates the solution based on the change in unfavourable resource consumption (such as fully assigned workstations or excessive workforce assignments). Quality robustness focuses on how well the value of the various quality metrics is retained after a disruption. Schedule stability refers to the ability of a schedule to remain consistent after disruptions.

Resource management evaluation

The effectiveness of the resource management of the model is evaluated using two methods. The first method evaluates how much additional excessive workforce had to be assigned when comparing the window in which the scenario is simulated between the start and end times. The results can be seen in figure 5.5.

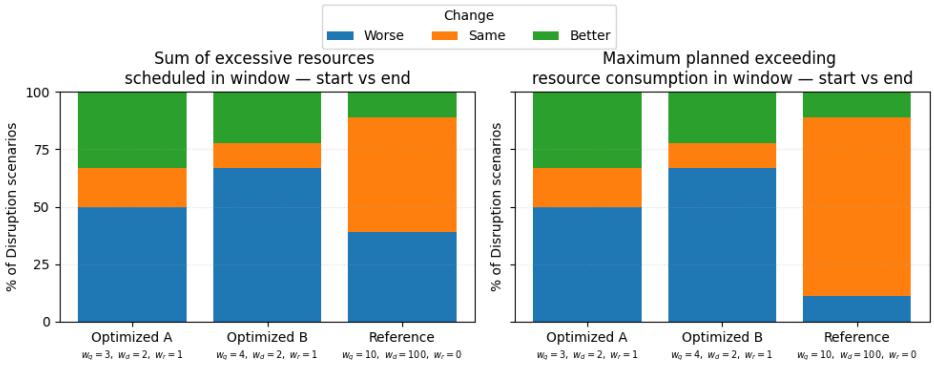
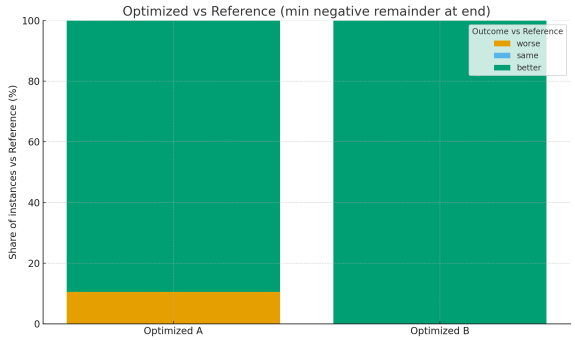


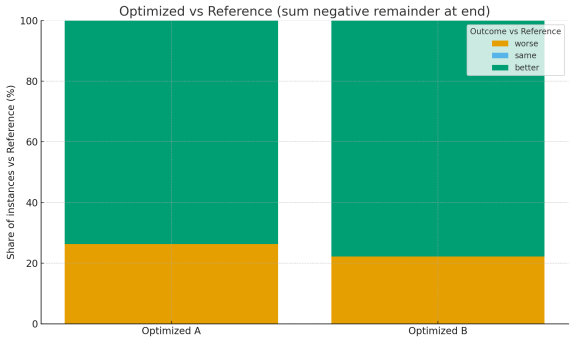
Figure 5.4: Comparison of the change in excessively planned resources before the disruption scenario occurred and once it has finished

The left plot shows for what percentage of the scenarios, the sum of excessively planned resources within the window incremented (worse), stayed the same or reduced (better). The plot on the right displays the percentage of scenarios in which the maximum value worsened, remained the same, or improved from the start to the end of the simulation.

When these results were found, they did not make any sense. How could a model that pushes to stay the same and finish quickly outperform a model that prepares itself for changes when comparing them on how well they can handle disruptions? Upon closer inspection, the plot made perfect sense. Whereas the optimised models were able to minimise excessive resource consumption, the reference model barely attempted to do so. So, if an optimised solution had to assign a couple of resources in excess, would that be nothing compared to how many resources the reference schedule already had planned in excess? If we compare the optimised solutions with the reference solution based on the excessively planned resource at the end of the scenario, then it suddenly tells a different story, as can be seen in figure 5.5



(a) Comparison of the excessively planned resources at the end of the simulation between the optimised solutions and the reference solutions.



(b) Comparison of the most excessively planned number of a resource at the end of the simulation between the optimised solutions and the reference solutions.

Figure 5.5: Side-by-side comparisons of excessive planning metrics at the end of the simulation.

In terms of resources, we can clearly see that robust rescheduling works significantly better than the reference schedule.

In terms of projects, the most important metric is how many projects suffered little to none from the introduced disruptions.

To measure the impact of the introduced disruption as accurately as possible, for each model in each scenario, the resulting schedule at the end of the disruption window was compared to the schedule found when no disruptions were introduced. Since a solving timeout was in place, comparisons may not always yield accurate results (using seeds for the solver and running on the hardware was implemented

	mean	median	std
Num. projects delayed beyond their reference actual TAT	0.111	0.000	1.231
$\sum_{i=1}^{N}(\mathrm{TAT}^i_{\mathsf{ref}} - \mathrm{TAT}^i_{\mathsf{cur}})$	-23.278	-7.500	54.009
Average delay in shifts for delayed projects	1.554	1.643	2.885
Average speedup in shifts for earlier projects	-7.333	-7.000	2.635

Table 5.6: Comparison of impact from disruptions between M_B and M_{REF} for N projects

to ensure results as accurate as possible). This is why the robust model $M_A = (w_q = 3, w_d = 2, w_r = 1)$ was left out of these results. Although the impact of disruptions seemed to align with M_B when comparing the starting schedule of the disruptive simulation with the ending state of the disruptive simulation, the comparison with the undisrupted reference data for the model was all over the place. After running the disruption scenario twice more, the results continued to vary, even with a larger solve time-out. Due to the lower weight on the quality objective, there are many more solutions with the same objective values, which explains the different results for each run. This is why M_A was left out of the following comparison.

Table **??** shows the compact results of the comparison between the impact of disruptions for M_B and M_{REF}. In this table, TAT_{ref} refers to the expected TAT of the scheduled projects, so not the contract TAT, and TAT_{cur} refers to the expected current TAT of the solutions. Since several projects were already weeks behind the contract timeline, it made no sense to compare them with projects that were expected to finish at their contract timeline.

The first metric describes the number of projects for which $\mathrm{TAT}^i_{\mathsf{ref}} < \mathrm{TAT}^i_{\mathsf{cur}}$. The second metric describes the sum of change in TAT_{cur} from M_{REF} to M_B, negative values indicate that M_B outperforms M_{REF}. The last two metrics describe the change for all delayed and sped-up projects separately.

More extensive information on the comparison can be found in appendix C

As can be seen, the optimised model unfortunately worsens slightly more overdue instances when compared to the reference model. However, this only occurs for 0.111 of the projects on average. When examining the actual differences across all projects, the disparity becomes significantly more substantial. Whereas the delay in delayed projects for the optimised model is, on average, less than two shifts more, the speedup of earlier projects is, on average, over seven shifts. Most importantly, as can be seen in C, the optimised model results in more sped-up projects in nearly all cases. In the *total shifts result*, it shows the net difference in shifts.

On initial evaluation of these results, they did seem relatively bloated and excessive. However, upon closer examination of the results, is this primarily due to the introduced fever window? As seen in A, a large buffer of projects is typically observed just outside the fever window. When a job is disrupted or some other change happens, the optimised model seems to move things in and out of the fever window and buffer. Utilising any available capacity to fit in other projects, or by moving a project from the fever window back to the buffer, if this would allow different projects to benefit from the introduced delay.

6

Conclusion

6.1. Conclusion

This thesis has demonstrated that using CP with weighted-sum multiple objectives, a rolling time horizon, and both proactive and reactive rescheduling is a reliable method for finding high-quality feasible solutions within short time spans.

RQ1 — Modelling the MRO as RCMPSP/max. We formalised the short-term Maintenance, Repair, and Overhaul (MRO) planning as a multi-project Resource-Constrained Project Scheduling Problem (RCPSP). This includes minimal and maximal time lags, time-varying capacities, and rolling-horizon rescheduling. The CP, using OR-Tools CP-SAT, incorporates precedence windows and limits on renewable resources such as workforce and workstations. Additionally, a deviation term is included in the objective to maintain stability between consecutive planning horizons.

RQ2 — Impact of robustness without disruptions.We compared the resulting schedules obtained by focusing on robustness and quality using the optimal model. Our findings indicate that while prioritising robustness slightly increases the time required to solve the scheduling problem, it leads to a minimal number of projects taking longer to complete. However, the use of a rolling fever score window **significantly improves resource consumption during that period**, as illustrated in the appendix. A

RQ3 — Proactive robustness under disruptions. When evaluating schedules disrupted by artificially generated disruptions, the optimal model's schedules would, in most cases, be able to perform more work at the cost of slightly worsening some projects. The fever window seems to add more flexibility to the environment by making sure that there are fewer limiting points in time for resources, allowing easier and less negatively impacting changes to the schedule after disruptions take place.

Answer to the main research question. The proposed CP model facilitates both proactive and reactive rescheduling in the target Maintenance, Repair, and Overhaul (MRO) environment. It is also able to outperform the other model during disruptions in terms of volume at a minimal cost of overdue TAT. Showing that exact methods can find high-quality robust solutions for the rescheduling of RCMPSP/max instances.

6.2. Discussion

6.2.1. Data quality and availability

Although the introduction of chapter 6.1 clearly shows the positive effects of the proposed proactive rescheduling model, it is still based on incomplete and flawed reference schedules. Not only do the reference schedules contain faulty data, but the intervals between reference schedules is also too long and inconsistent.

Also, to find more robust workforce assignments, the model sets bounds on the required FTE for each job during each shift. In reality, are these jobs made up of a lot of smaller jobs (1.6.4). Meaning that in fact, the assigned number of FTE is based on the sum of the required FTE of all sub-jobs scheduled

in a shift. Splitting these sub-jobs over multiple shifts is usually avoided in reality. Thus, the flexibility of employee assignments would likely be significantly worse if more detailed information about the environment had been available.

Another issue stemming from incomplete information is the incomplete resource requirements. The only documented resource requirements are a few authorisations, the engine stands and the test cell. A few examples of resources that are not included are the module workstations, *tooling* (carts used to move modules between engine stands and module workstations) and the core wells for the Genx engines.

Even if these resources were included in the input data, it would still be nearly impossible to adequately compare solutions with reference schedules, as the latter often violate their own constraints, which means that the provided input data, which should be used for reference, is not a valid source.

6.2.2. Model Enhancements

Currently, the fever function is a relatively simple function that ends up moving jobs to the edge of the rolling fever window (as can be seen in A) and spreads them out from there. I expect the model to perform even better with even fewer deviations if the window ended with a ramped-down fever score. This would hopefully trigger a reschedule for jobs that would otherwise likely be rescheduled once they reach the rolling time window.

6.2.3. Reliability of results

As mentioned in 5.3.2, one model was excluded from the evaluation due to the high variance in the results. For any future work or verification, I would suggest that the benchmark for these results should not only be performed with an even larger solve timeout but also by deriving several reference solution baselines for each model. This should minimise the odds of comparisons between entirely different schedules from the same models.

6.3. Recommendations

Before proceeding with any further work on this topic, I recommend focusing on ensuring data consistency and availability. The results of the research should be more reliable and consistent if all the gaps are filled. Otherwise, I would recommend exploring what a more advanced fever score and fever window can achieve. Currently, the function is set to a simple method to prevent the problem from becoming even harder. But if more information about the environment is known, more complex fever score functions and windows might be able to improve the robustness even more, costing even less of its quality

References

[1] *Anago — Go for impact*. Anago. URL: https://www.anago.io/nl (visited on 10/07/2025).

[2] JaeBong Cho et al. “Efficient Task Scheduling Using Constraints Programming for Enhanced Planning and Reliability”. In: *Applied Sciences* 14.23 (2024). Published 6 December 2024, p. 11396. DOI: 10.3390/app142311396. URL: https://doi.org/10.3390/app142311396.

[3] S. Hartmann and D. Briskorn. “An updated survey of variants and extensions of the resource-constrained project scheduling problem”. In: *European Journal of Operational Research* 297.1 (2022), pp. 1–14. DOI: 10.1016/j.ejor.2021.05.004. URL: https://www.sciencedirect.com/science/article/pii/S0377221721003982.

[4] Sönke Hartmann and Dirk Briskorn. “A survey of variants and extensions of the resource-constrained project scheduling problem”. In: *European Journal of Operational Research* 207.1 (2010), pp. 1–14. DOI: 10.1016/j.ejor.2009.11.005.

[5] Tim Hepher. *Aircraft engine maintenance times at historic high, Bain says*. July 2024. URL: https://www.reuters.com/business/aerospace-defense/aircraft-engine-maintenance-times-historic-high-bain-says-2024-07-19/ (visited on 10/07/2025).

[6] Willy Herroelen and Roel Leus. “Robust and Reactive Project Scheduling: A Review and Classification of Procedures”. In: *International Journal of Production Research* 42.8 (2004), pp. 1599–1620. DOI: 10.1080/00207540310001638055. URL: https://doi.org/10.1080/00207540310001638055.

[7] International Air Transport Association (IATA). *Global Outlook for Air Transport: Protectionism on the Rise*. IATA Sustainability and Economics. Montreal, Quebec, Canada: International Air Transport Association, June 2025, p. 35. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/global-outlook-for-air-transport-june-2025/ (visited on 10/07/2025).

[8] International Air Transport Association (IATA). *How Many Aircraft Are We Missing?* IATA Economics Chart of the Week. IATA Economics, “How many aircraft are we missing?”, 20 June 2025, report available from IATA’s website. June 2025. URL: https://www.iata.org/en/iata-repository/publications/economic-reports/how-many-aircraft-are-we-missing/.

[9] International Air Transport Association (IATA). *Industry Statistics Fact Sheet*. IATA Global Outlook for Air Transport—Industry Statistics. PDF fact sheet, updated June 2025. Includes data on global airline industry indicators such as flights (million), passengers, revenues, etc. June 2025. URL: https://www.iata.org/en/iata-repository/pressroom/fact-sheets/industry-statistics.

[10] Philippe Laborie et al. “IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG”. In: *Constraints* 23 (Mar. 2018). DOI: 10.1007/s10601-018-9281-x.

[11] Leon Lan and Joost Berkhout. “PyJobShop: Solving scheduling problems with constraint programming in Python”. In: *arXiv preprint arXiv:2502.13483* (2025). DOI: 10.48550/arXiv.2502.13483. URL: https://arxiv.org/abs/2502.13483.

[12] Laurent Perron and Frédéric Didier. *CP-SAT*. Version v9.12. Google, Feb. 17, 2025. URL: https://developers.google.com/optimization/cp/cp_solver/.

[13] Arno Sprecher, Rainer Kolisch, and Andreas Drexl. *Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation*. Working Paper. Christian-Albrechts-Universität zu Kiel, 1994. URL: https://www.econstor.eu/bitstream/10419/155418/1/manuskript_344.pdf.

[14] Nguyen Thi Ngoc Truc et al. “Buffer Management Method for Multiple Projects in the CCPM-MPL Representation”. In: *Industrial Engineering & Management Systems* 11.4 (2012), pp. 397–405.

[15] World Bank, World Development Indicators. *Air Transport, Passengers Carried*. World Bank World Development Indicators. Includes domestic and international passengers carried by air carriers registered in each country; data originally sourced from ICAO's Civil Aviation Statistics of the World and ICAO staff estimates. CC BY-4.0 license. 2025. URL: https://data.worldbank.org/indicator/IS.AIR.PSGR.

[16] World Bank, World Development Indicators. *Air Transport, Registered Carrier Departures Worldwide*. World Bank World Development Indicators. Represents total scheduled flight departures (take-offs) — domestic and international — of air carriers registered in each country; sourced from ICAO Civil Aviation Statistics of the World and ICAO staff estimates. CC BY-4.0 license. 2025. URL: https://data.worldbank.org/indicator/IS.AIR.DPRT.

A

Example of the rolling time window for the fever score

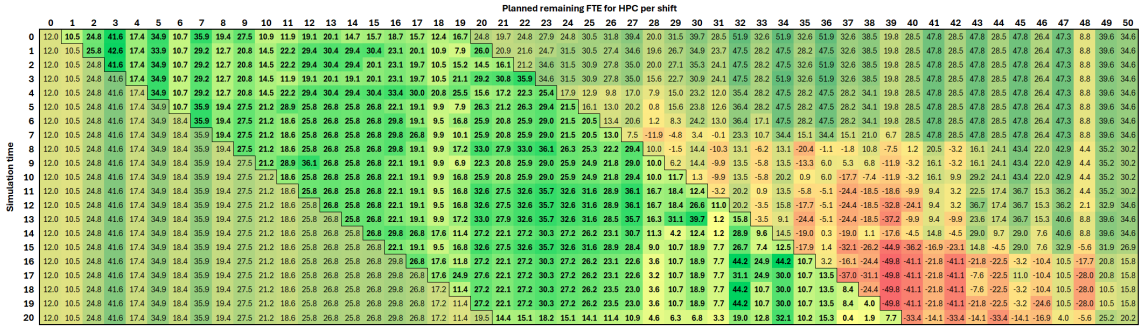


Figure A.1: A visualised subset from the fever scores given to an instance of the optimised model. Cell borders were used to indicate the rolling fever window, as well as a slightly brighter colour scale within the rolling fever window. One can clearly see the fever scores increasing outside the window, but never entering the actual window. Which corresponds to the

B

Metrics for various combinations of hyperparameters

Table B.1: All parameter combinations for the Optimal series (early window 0–20, late window 60–71).

w_q	w_d	w_r	Fever	Joined	DynDur	DynFTE	Mean # unchanged	Mean # to do	Mean net avg	Late #
1	1	0	False	False	False	False	49.2500	18.6944	-0.0370	
10	10	0	False	False	False	False	49.2500	18.6944	-0.0370	
10	1	1	True	False	True	True	49.2222	18.4306	-0.0278	
1	10	1	True	False	False	False	49.2639	18.7361	-0.0972	
1	10	0	False	False	False	False	49.2639	18.7361	-0.0972	
10	10	1	True	True	True	True	48.6944	18.5556	0.2344	
10	1	1	True	True	True	True	49.1250	18.4167	-0.1620	
1	10	0	False	True	False	False	49.2222	18.6944	-0.1898	
10	10	0	False	True	False	False	49.2222	18.6944	-0.1898	
1	10	1	True	True	False	False	49.2222	18.6944	-0.1898	
1	1	0	False	True	False	False	49.2222	18.6944	-0.1898	
10	10	1	True	False	False	False	49.0417	18.5278	0.1181	
10	1	1	True	False	False	False	49.1806	18.5139	-0.1366	
1	1	1	True	False	False	False	49.1250	18.5833	0.0000	
10	1	0	False	False	False	False	48.8889	18.6944	0.4560	
10	1	1	True	True	False	False	49.1944	18.6111	-0.1713	
10	1	0	False	True	False	False	48.8889	18.7083	0.2616	
1	10	0	False	False	True	True	49.1389	18.8472	-0.2500	
10	1	0	False	True	True	True	48.7083	18.6250	0.0090	
10	10	1	True	True	False	False	49.1250	18.7639	-0.1667	
1	10	1	True	False	True	True	48.6667	18.4028	0.0174	
1	1	1	True	True	False	False	49.1111	18.7639	-0.1667	
10	10	1	True	False	True	True	48.5278	18.5139	0.1449	
10	1	0	False	False	True	True	48.6389	18.4444	-0.2000	
10	10	0	False	False	True	True	48.9861	18.5694	-0.2072	
1	1	1	True	True	True	True	48.6389	18.8194	-0.0359	
1	1	0	False	False	True	True	48.9722	18.5694	-0.4190	
1	1	1	True	False	True	True	48.3194	18.7083	0.0519	
1	1	0	False	True	True	True	48.9722	18.6250	-0.3991	
10	10	0	False	True	True	True	48.9444	18.9722	-0.7847	
1	10	1	True	True	True	True	48.6111	18.8611	-0.7222	
1	10	0	False	True	True	True	48.9722	19.1944	-0.6319	

C

Full comparison of model statistics during simulations introducing disruptions

Table C.1: Optimized − Reference deltas per scenario (full dataset).

Scenario	# later project ends	# unchanged ends	# earlier project ends	Total shifts lost	Total shifts won	Total shifts result	Later than reference actual TAT	Sum reference actual TAT	Average delay for delayed projects	Average speedup for earlier projects
1.0	5.0	-9.0	5.0	32.0	-95.0	-63.0	1.0	-25.0	3.666666667	-9.0
2.0	3.0	-6.0	4.0	31.0	-31.0	0.0	0.0	9.0	5.0	-6.0
3.0	1.0	-12.0	17.0	11.0	-89.0	-78.0	1.0	-8.0	4.5	-12.0
4.0	4.0	-7.0	3.0	7.0	-5.0	2.0	1.0	2.0	0.0	-7.0
5.0	-4.0	-7.0	10.0	-12.0	-418.0	-430.0	-1.0	-222.0	0.8	-7.0
6.0	2.0	-7.0	3.0	21.0	-38.0	-17.0	-2.0	-23.0	3.8	-7.0
7.0	2.0	-5.0	3.0	0.0	-12.0	-12.0	0.0	-3.0	-1.2	-5.0
8.0	2.0	-8.0	-4.0	-4.0	-9.0	-13.0	-2.0	-7.0	-2.228571429	-8.0
9.0	-2.0	-6.0	4.0	-2.0	-218.0	-220.0	0.0	-73.0	0.8	-6.0
10.0	4.0	-11.0	8.0	18.0	-61.0	-43.0	1.0	-18.0	1.619047619	-11.0
11.0	2.0	-11.0	10.0	20.0	-27.0	-7.0	1.0	2.0	4.25	-11.0
12.0	-1.0	-6.0	13.0	-1.0	-66.0	-67.0	-1.0	-39.0	-1.0	-6.0
13.0	3.0	-5.0	2.0	5.0	-2.0	3.0	2.0	3.0	1.666666667	-5.0
14.0	1.0	-4.0	2.0	22.0	-36.0	-14.0	0.0	27.0	2.714285714	-4.0
15.0	-2.0	-9.0	9.0	14.0	-72.0	-58.0	-1.0	-25.0	4.208333333	-9.0
16.0	2.0	-2.0	0.0	7.0	-5.0	2.0	1.0	-3.0	1.25	-2.0
17.0	1.0	-7.0	-4.0	-22.0	6.0	-16.0	-1.0	-11.0	-6.0	-7.0
18.0	2.0	-10.0	4.0	42.0	-63.0	-21.0	2.0	-5.0	4.125	-10.0