Vrije Universiteit Amsterdam

Lynxx

Master Thesis

# Models and Methods for Optimizing Alternative Train Timetables

**Author:** Lara Owobowale (2704852)

*1st supervisor:* Rob van der Mei
*daily supervisor:* Robin Weber (Lynxx)
*2nd reader:* René Bekker

*A thesis submitted in fulfillment of the requirements for the VU Master of Science degree in Business Analytics*

July 9, 2026

*"I am the master of my fate, I am the captain of my soul"*
*from* Invictus*, by William Ernest Henley*

## Management Summary

The generation of alternative train schedules during planned infrastructure maintenance is highly complex. This thesis optimizes the interaction between macroscopic network planning and microscopic station safety validation within ProRail's scheduling framework to drastically reduce computation times.

**1. The Problem** The current scheduling model relies on an iterative loop where the macro-layer creates network schedules, but the micro-layer must blindly check every station changed by the macro model for local capacity conflicts, creating a severe computational bottleneck. Data exploration reveals massive inefficiencies in this design, showing that 70% to 80% of these microscopic computations are completely redundant because those stations are already conflict-free. This blind validation loop causes high overall computation times, preventing ProRail from generating alternative timetables quickly enough to meet real-time operational demands.

**2. The Results** To address this challenge, this thesis proposes a "Filter, Rank, and Stop" framework that filters out guaranteed safe stations, ranks the remaining stations by conflict risk using statistical and machine learning models, and dynamically stops the queue early once all active conflicts are identified. This solution eliminates up to 63% of unnecessary micro-level computations and slashes total system execution time by more than 75% through sequential, dynamic feature compilation. These findings present ProRail with two distinct implementation choices depending on organizational priorities: a risk-averse method that ensures zero missed conflicts or a time-efficient method designed to maximize processing speed.

**3. Operational Recommendations** For scenarios demanding maximum safety, ProRail should deploy a risk-averse profile combining the machine learning ranker with a *Thompson Posterior* stopping rule. This configuration minimizes missed infrastructure conflicts to an absolute minimum and skips up to 55% of the full iteration. Alternatively, for an efficiency-focused profile maximizing time savings, the network should deploy the *Statistical Sequential Probability Ratio Test (SPRT)* stopping rule, which shortens the verification loop at the earliest statistically justifiable moment. Finally, removing two features that take a long time to build allows the model to reduce the iterations even further, decrease the baseline runtime by 75%.

# Contents

# 1

# Introduction

In the early days of railroads, scheduling was straightforward enough to be handled manually. With networks comprising only a few stations and limited connections, planners could easily create timetables with just a pencil and paper, often simply repeating train departures at an hour.

However, today's approach cannot rely on manual methods. The scheduling challenge has become significantly more intricate, involving hundreds of stations and dense networks of interconnected tracks. The most complex version of this challenge is found in the Netherlands, which has the busiest rail network in the European Union, averaging more than 20,000 train kilometers per kilometer of track annually (1). As networks have expanded, modern railway systems must adhere to strict regulations and safety protocols that dictate when and where trains can use intersecting infrastructure. Furthermore, the different types of trains with different speeds allow overtaking, which adds flexibility but complicates scheduling (1). The scheduling process must account for several operational constraints, including platform lengths, safety margins between trains, and specific residing times. Mathematically, these constraints are based on accurate tracking of the time between trains arriving on the same platform, a framework established formally through the Periodic Event Scheduling Problem (PESP) (2) and extensively studied in various rail transit systems (3). Given that the complexity of this problem far exceeds human capacity, advanced algorithmic solutions have become essential, compounded by the NP-hard combinatorial challenge of computing cyclic railway schedules across large interwoven networks (4, 5).

Moreover, a railway network requires continuous maintenance to repair tracks, renovate stations, or remodel platforms. While these repairs take place, specific tracks or stations are temporarily blocked. During these periods, an alternative timetable is required to

operate on the reduced network. As illustrated in Figure 1.1, finding this alternative timetable is highly complex; while a macro-level adjustment might seem straightforward on a network map, it must simultaneously respect the tight, low-level station capacity constraints, platform allocations, and interlocking tracks shown below. Deciding which maintenance tasks can be combined and when they should take place depends heavily on the alternative timetables the model can generate. Importantly, any alternative timetable must stay as close to the original schedule as possible. Completely rewriting a schedule causes disruptions for train crews and station staff. Since maintenance happens frequently, creating a new schedule every time would lead to irregular and unmanageable working schedules.

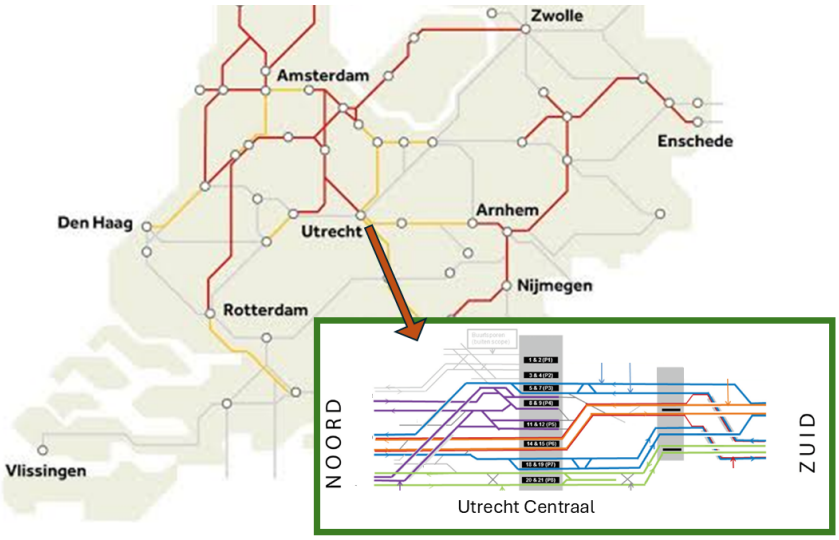


**Figure 1.1:** A visual contrast of scheduling dimensions: the macroscopic network layer managing route deviations during disruptions (top) versus the dense structural complexity of track and platform interlockings at the individual station level (bottom).

To solve this, ProRail (the Dutch railway infrastructure manager) tasked Lynxx, in collaboration with its internal RAAD department, to develop an automated optimization model capable of generating these alternative timetables (6). They have created a model that divides the rescheduling process into two parts: a macro network layer and a detailed micro layer. The macro model passes a list of stations down to the micro level. The micro algorithm then tries to make the station conflict-free. If this is impossible, it returns constraints to the macromodel to be included in the next iteration. These models iterate consecutively until both layers are feasible and a completely conflict-free schedule is found.

## Problem Statement

Despite mathematical advances in dual-layer rail optimization, a severe operational bottleneck remains in the communication loop between macroscopic planning and microscopic validation. Currently, the macro model passes every modified station down to the micro layer for safety and capacity verification. Because this process lacks predictive foresight, the system blindly performs expensive microscopic computations on all transferred stations.

Data exploration reveals that between 70% and 90% of these micro-computations are entirely redundant because those stations are already mathematically conflict-free. This massive volume of wasted processing cycles causes slow, exhaustive iterative loops, rendering the execution time too high for real-time dispatching and agile maintenance planning. Therefore, a targeted framework is required to isolate critical infrastructure friction points and eliminate unnecessary micro-level validations without risking information leakage.

## Research Focus and Methodology

To address this clear computational bottleneck, this research focuses on improving the communication and data exchange efficiency between the two architectural layers. The main research question is:

> *"How can the communication between the macro layer and micro layer be enhanced to identify the most critical conflict areas and improve overall model efficiency?"*

To answer this, this thesis proposes a "Filter, Rank, and Stop" method that aims to reduce the number of wasted micro computations and create a more dynamic communication between the two layers. This method takes the list of stations from the macro layer and ranks them by difficulty, from hardest to easiest to solve. The stations are then processed in that specific order. Additionally, a stopping mechanism is introduced to stop the process when only stations remain that are easily solvable by the micro model and do not generate any new constraints. By stopping early, many redundant micro computations are skipped, which reduces both the iteration time and the overall execution time of the total model.

The remainder of this research is structured as follows. First, Chapter 2 introduces the company background and the specific project framework established by ProRail, followed by a detailed problem description in Chapter 3. Chapter 4 provides a comprehensive literature review of relevant scheduling methodologies. Next, the current model used by Lynxx is examined in detail in Chapter 5, while Chapter 6 outlines the data set and

provides an exploratory data analysis. Chapter 7 then explains the proposed Filter, Rank, and Stop method. After this, the experimental setup is detailed in Chapter 8, followed by the presentation and analysis of the results in Chapter 9. Finally, the findings are summarized in the conclusion, and steps for future research are outlined in the discussion section.

# 2

# Project and Company


RAAD, short for *Rekenen aan Alternatieve Dienstregeling*, is a project executed by ProRail and Lynxx. ProRail is the organization that manages rail infrastructure in the Netherlands. They take care of the maintenance, renewal, expansion and safety of a 7,000 km railway network and more than 400 stations. A key job for ProRail is to assign track capacity and time slots to more than 30 passenger and freight operators that use the railway network (1). The Dutch baseline timetable demonstrates that busy networks cannot expand capacity without advanced automated scheduling models (7). Since passenger services and slower freight trains often share tracks, planning involves complex interactions that can lead to infrastructure conflicts (1). To effectively manage these situations, specific mathematical solutions are needed to integrate freight routes without causing disruptions in passenger services (8). The RAAD department focuses on managing track capacity and logistics, especially when maintenance or other interruptions require an alternative timetable.

To develop the RAAD model, ProRail collaborated with Lynxx. Lynxx is a data science and operations research consulting firm that provides technical solutions to complex business problems. The company was inspired by the large amount of public transport data from the national OV-chipkaart system and was established to help transport companies make sense of their data. Lynxx has grown into a specialized firm with over 40 employees focusing on transport and logistics, with significant industry experience (9). A dedicated team from Lynxx has worked for two years to create the complex model needed for this scheduling task.

ProRail set clear goals for the Lynxx team. The RAAD model aims to automatically create an alternative schedule that meets safety standards, minimizes inconvenience for passengers, and remains operationally feasible. An important goal is to keep the changes to a minimum compared to the original schedule. This helps reduce passenger dissatisfaction and reduces the effort needed to adjust crew and vehicle schedules (10). This challenge is a classic multi-objective optimization problem, where a framework must evaluate trade-offs between passenger inconvenience, operational costs, and changes to the original schedule (11, 12). An important reason for the innovation for this model is the European Time Table Redesign (TTR) program (13). Time Table Redesign (TTR) is a European initiative that improves railway scheduling and capacity management across borders. Starting with the 2031 schedule, ProRail must comply with the new standardized steps introduced by TTR. ProRail needs to update its systems and has stricter deadlines for timetables. ProRail has to provide the timetable 18 months in advance, including alternative tables for maintenance periods (14). The Lynxx mathematical model, developed in the RAAD project, supports this transition by automating the creation and assessment of compliant timetables

# 3

# Problem description

To create the alternative train timetables, the Lynxx team has developed a two-layer framework: **macro-level** and textbfmicro-level. The **macro-level** focuses on the network reroute planning, treating stations as nodes to optimize service during maintenance (1, 15). This approach is effective for long-term planning and network optimization (16). Meanwhile, the **micro-level** deals with detailed station constraints, managing track sections and signal timings to avoid capacity violations (15, 17, 18). It assigns tracks and platforms, preventing conflicts at crossover points.

These levels work in an iterative feedback loop similar to a decomposition scheme (15). They adjust network assignments and headway restrictions until a coherent schedule is created (19). If conflicts occur at the micro-level, they inform the macro-level, which re-optimises the schedule to resolve these issues using guided adjustments (5). This process continues until a feasible and conflict-free schedule is established.

Despite this two-layer method providing a solid framework, it could be further improved. The communication between the two layers can be enhanced by only computing micro algorithms on conflicted areas instead of computing each station regardless of its status. Beyond just reducing computation times, this targeted approach helps identify the most promising "neighborhoods" at the microlevel. By focusing on these specific areas, the model can discover the best information to communicate back to the macrolevel, leading to faster and higher-quality improvements. Scheduling problems are complex and often have too many constraints, making it hard to find feasible schedules quickly (4). Consequently, the long computation times prevent real-time planning for infrastructure managers (20). *Therefore, the core objective of this thesis research is to optimize the current model to achieve a significantly faster computation time by refining macro-micro communication.*

# 4

# Literature Review

## 4.1 Difficulties in Periodic Railway Timetabling

Developing an optimization algorithm for a railway infrastructure is challenging due to the need to integrate fixed track layouts and safety signaling into a cohesive mathematical model.

First, passenger train schedules are periodic, which means that train departures occur cyclically throughout the day, typically repeating every hour (1). However, each train movement follows its own cycle, and some regional lines operate on two-hour or longer cycles, creating a collection of different cycle lengths that conceal the underlying periodicity. The mathematical challenge is enhanced by the cyclic nature of time, which requires the use of discrete integer counters to determine the timing of train events within these cycles (10).

Furthermore, real-world conditions increase the complexity even more. A perfectly symmetric schedule is beneficial for both passengers and operators, but limitations such as the physical infrastructure of the network make this impossible. In some cases, there are differences in the direction of the tracks due to slopes or localized curves that limit the speed in one direction. Kummling et al. therefore state that train running times are inherently asymmetric (4). Consequently, modern mathematical formulations must explicitly integrate this directional asymmetry by incorporating a special symmetry deviation constraint to allow a margin of slack ($d_a$) within the network. In this formulation, $T_i$ and $T_j$ represent the scheduled times for the arrival and departure events of the associated opposing trains, $s$ is the network-wide symmetry minute (axis), $T$ is the total period duration of the timetable, and $z_a$ is an integer parameter used to adjust for the periodic nature of the

schedule. This relationship is formally defined as follows:

$$2s - 2d_a \leq T_i + T_j - z_a T \leq 2s + 2d_a \tag{4.1}$$

Liebchen et al. introduce another demand that the schedule must fulfill (21). When planners create the schedule, they must also adhere to operational hierarchies. There are high-priority services, such as long-distance trains, that are optimized first, creating fixed schedules that limit the flexibility for regional trains. International services introduce even stricter timing requirements. Each international train has border times at which it has to cross, giving very little room for adjustment by planners (1).

Moreover, the disconnect between the planning layers could give a feasible schedule on macro level but fail on micro level. At the micro level, there are safety measures in place that take into account the dynamics of the train. For example, the braking distance of a train is used to determine the safety norm of when a train is allowed to enter the platform after another has left. These margins are formulated by sequencing equations that make sure that the train has left the platform. However, in very dense networks, this problem is further complicated by the "blocking or hold-while-wait" constraints; due to a lack of intermediate buffer space, a track section cannot release a train and has to keep it at the platform until the next sequential block on its route becomes entirely clear (22). Managing these sequential blocks for all the varying speeds of the trains introduces even more complexity. Sprinters accelerate rapidly but stop frequently, whereas intercity trains accelerate slowly but then maintain high speed (1). In some advanced contexts, models must even take into account the elasticity of the train length itself, compressing or stretching train configurations to accommodate the limits of the route (23). If trip time windows are flexible, standard headway bounds are not enough to ensure that a fast intercity train will not illegally overtake a slower freight train on the same line (21). Lastly the stations also differ in layout, platform lengths and capacity caps (24, 25). Most conflicts do not happen on open tracks; they happen before a large station in the throat area where several paths cross to access designated platforms (1).

## 4.2 Complexity and Proof of Hardness

To understand why optimizing alternative train schedules is so difficult, it is essential to examine the mathematical foundation of the problem itself. The periodic timetabling problem in Operations Research (OR) is classified as NP-complete, which means that it is computationally challenging. Problems in Class P can be solved quickly, while those

in Class NP can be verified quickly. NP-complete problems are the most difficult in NP, as any NP problem can be transformed into an NP-complete problem in polynomial time (26).

The complexity of periodic timetabling has been established through research by Serafini et al. (2), showing that it can be reduced from classic problems like the Hamiltonian Circuit Problem. Odijk reduced the PESP problem to the Graph Coloring Problem (27). While determining if a valid periodic schedule exists is NP-complete, generating a high-quality schedule is NP-hard. This complexity increases when multiple objectives are involved, such as minimizing passenger travel time and operational costs (5). Such multi-objective optimization requires advanced heuristics and cannot guaranty a polynomial running time, as it explores trade-offs in a complex solution space. Binder et al. explore the multi-objective case, using epsilon constraints and by constructing a Pareto Frontier (11).

## 4.3 Modern Algorithmic Models and Decomposition Methods

To manage the complexity of modern railway networks, most new approaches use decomposition methods or iterative heuristics to navigate to the optimal schedule. One of the known approaches is hierarchical decomposition, which is inspired by the priority train type method used by planners. It optimizes the long-distance networks first, then moves on to local and freight layers. Advanced sequential decomposition techniques solve a sequence of smaller parametrizable subproblems, finding a middle ground between global scheduling and rigid sequential planning (20).

Other strategies use local conflict search and corridor analysis. Instead of processing a complete network, these algorithms only take specific nodes and arcs that are directly impacted by maintenance and expand slightly into neighbouring areas. These clusters are then solved independently and inserted back into the global network.

An innovative approach to railway optimization combines macroscopic and microscopic models into automated systems, demonstrated by Logic-Based Benders Decomposition (15). In this framework, a master macroscopic problem controls the global train routing, while microscopic subproblems focus on safety and capacity at a detailed level. This ensures that running and headway times are managed at a macro level without layout conflicts when reverted to microscopic dimensions. The master problem sends proposed event times to subproblems, which check for infeasibility (like collisions or violations) and generate combinatorial Benders cuts to return to the master problem. This process can

also be structured as a two-stage solution or through time-discretized networks with AI optimization (18).

To reduce computational load, modern Benders implementations include change-detection logic, skipping redundant evaluations when macro inputs are unchanged. Goerigk et al. provides a way to escape local optima can be to have algorithms alternate between a heuristic network and the modulo network simplex approach (ModSim) (28). Running these two methods alternately allows one procedure to find an improving pathway out of a local optimum precisely when the other subprocedure has stalled.

To handle the heavy computation requirements of these multi-layered iterations, modern frameworks rely on parallel computing approaches, such as message passing interfaces (MPI), to execute sub-processes simultaneously (29) (30). Additionally, distributed frameworks can divide the constraint network into localized tree structures, using a distributed and asynchronous search (DTS) algorithm where agents use a 'Nogood_message' to aggressively prune the search space when conflicts arise (31).

# 5

# The Baseline Macro-Micro Scheduling Framework

This chapter explains the mathematical structure of the scheduling system. The specific dual-layer architecture described here was developed and implemented by Lynxx. The model operates as a multi-level optimization framework that balances network-wide macro planning with localized microscopic capacity validation (15, 19), forming the baseline system that this research aims to optimize.

## 5.1 System Architecture Overview

Generating a conflict-free timetable that is feasible on both the network and station level is a major computational challenge. If a model attempts to solve both levels at the same time, each station would add its own complex set of capacity, safety, and interlocking constraints. Given that the Netherlands contains hundreds of stations, combining all these requirements into a single model would create too many constraints. This massive scale makes it nearly impossible for a search algorithm to find a solution. Therefore, the Lynxx team has chosen to divide the problem into two layers that interact with each other, an approach that is often used for such complex problems (15):

1. **The Macroscopic Layer:** Formulates a network-wide alternative timetable using a linearized Periodic Event Scheduling Problem (PESP) structure to execute train routing, line interventions, and global time-shifting (2, 10). Each *Dienstregelpunt* (DRP) is visualized as a node and the connections between stations are considered as edges (16). *Dienstregelpunten* (DRPs) are stations, bridges, switch points and other parts of shared infrastructure.

2. **The Microscopic Layer:** Implements a detailed Station Capacity Model (SCM) that evaluates the feasibility on track-level. It ensures that trains navigate platforms, shared crossover points, and switch throats safely based on fixed macroscopic timings (15, 25).

This iterative bottleneck is illustrated in the macro-micro feedback loop in Figure 5.1. As shown, when the macroscopic PESP layer modifies schedules during disruptions, the microscopic SCM layer must run separate capacity computations for every changed station, which results in a high volume of calculations even though not every DRP generates an active constraint.

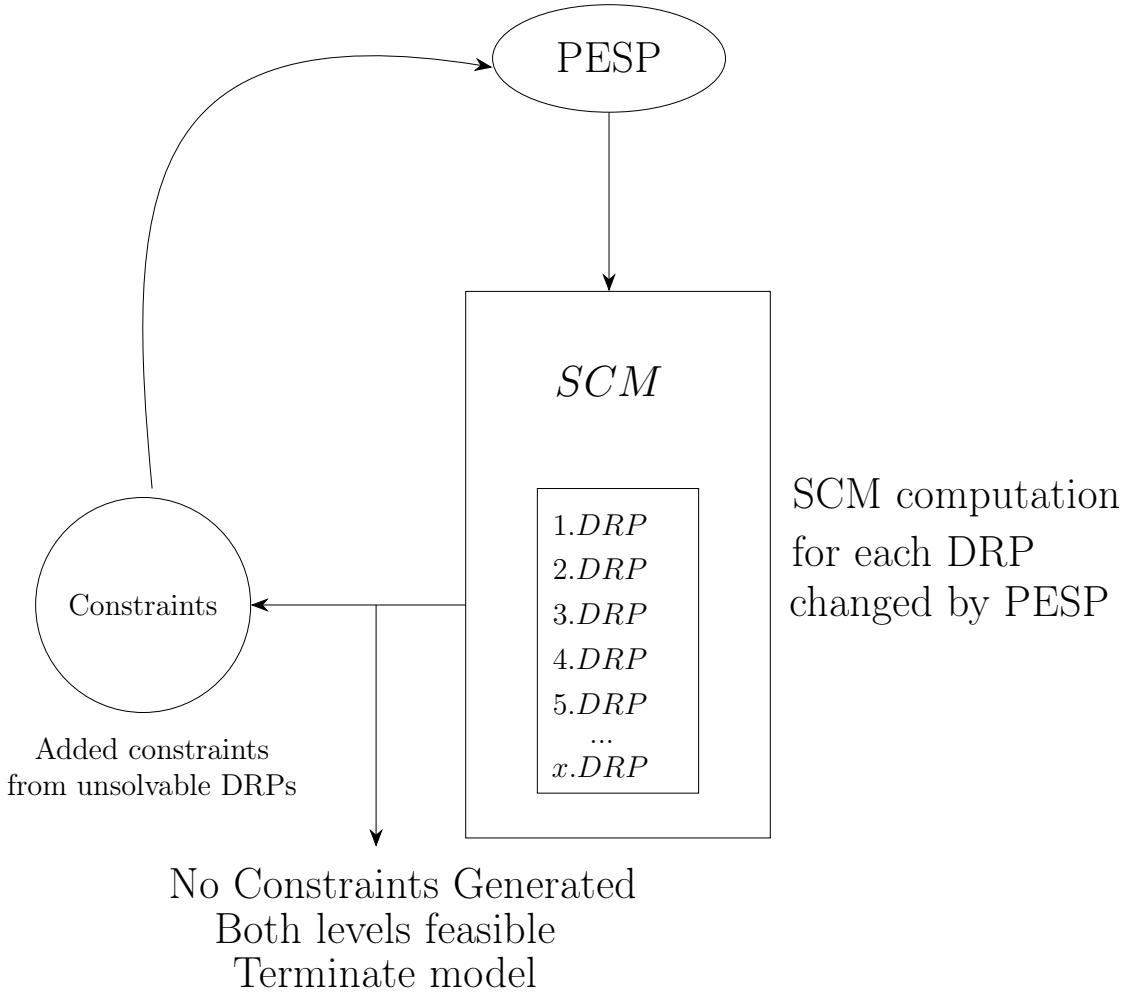


**Figure 5.1:** The iterative macro-micro feedback loop showing automatic constraint generation between the PESP planning layer and the local SCM validation system.

The macroscopic algorithm has four different objectives that it aims to optimise (11, 12):

- **Stakeholder Equity:** The distribution should be somewhat equal between the freight operators ($M^G$) and the passenger operators ($M^R$), to ensure that the impact of maintenance does not fall disproportionately on a single transport sector(7, 8).
- **Passenger Convenience:** Train cancellations are penalized and it prioritizes standard preferred turnarounds (*voorkeurskeringen*) to minimize global passenger travel hindrance (5, 11).
- **Timetable Robustness:** Maximizes operational stability by punishing tight schedule transitions, guaranteeing that sufficient buffer times are distributed between subsequent train paths to prevent minor delays from cascading (19, 32).
- **Operational Feasibility:** Restricts schedules based on resource constraints, ensuring that conductors have enough walking time during vehicle turns (*keringen*) and securing that adjusted paths remain within the available driver pool (7, 21).

## 5.2 The Macroscopic Layer: Periodic Event Scheduling Problem Model

The macroscopic optimization algorithm uses a specialized mathematical structure to plan changes in the timetable over a repeating 24-hour cycle ($T = 1440$ minutes). By extending the foundational Periodic Event Scheduling Problem (PESP), the model defines an alternative timetable using the tuple $(v, q, y)$, where $v_i \in [0, T-1]$ represents the new periodic event times, $q_{i,j} \in \{0, 1\}$ determines the order of events in the alternative timetable, and $y_{m,f} \in \{0, 1\}$ tracks cancellations due to infrastructure maintenance blocks (2). For each active network activity arc $a = (i, j)$, an additional decision variable $\alpha_i \in \{-1, 0, 1\}$ tracks cycle boundary crossings (10, 21). To allow trains to partially restart after a mid-route cancellation, the model introduces a binary restart variable $z_{m,f}$ and an integer tracker $\gamma_m$. In addition, two parameters represent the baseline schedule to measure deviations: the original event time $\pi_i \in [0, T-1]$ and the original event order $p_{i,j} \in \{0, 1\}$.

### 5.2.1 Sets and Activity Graph

The network and timetable are modeled as an Event-Activity Graph $G = (E, A)$, spanning the following core components:

- $s \in S$: The set of timetable points (*Dienstregelpunten*), such as stations, junctions, and bridges.

- $m, n \in M$: The set of train movements (*Treinbeweging*), partitioned into passenger trains ($n \in M^R$) and freight trains ($m \in M^G$). Freight trains consist of a single segment, whereas passenger trains are divided into segments $f, g \in Seg_m$ separated by predefined turnaround locations (*keerlocaties*).

- $i, j \in E$: The set of events $i, j$ (e.g., arrival, departure, passing, or split short-stops), where $E_m^f \subset E$ represents the events belonging to segment $f$ of train $m$.

- $(i, j) \in A$: The set of activity arcs $(i, j)$ tracking required time intervals. This includes internal train operations ($A^m = A^m_{\text{rijden}} \cup A^m_{\text{korte stop}} \cup A^m_{\text{langestop}}$) and multi-train safety or routing interactions ($A^{m,n} = A^{m,n}_{\text{headway}} \cup A^{m,n}_{\text{frontaal}} \cup A^{m,n}_{\text{kruising}} \cup A^{m,n}_{\text{kering}}$).

- $A^{m,n}_{\textbf{kering}}$: Turnaround activities, categorized either by type ($A^{m,n}_{\text{reizigers}} \cup A^{m,n}_{\text{mixed}} \cup A^{m,n}_{\text{LM}} \cup A^{m,n}_{\text{dummy\_kering}}$) or by data source ($A^{m,n}_{\text{voorkeurskering}} \cup A^{m,n}_{\text{alternatievekering}}$). Turnaround sets are explicitly separated by train combinations into $A^{m^R,n^R}$, $A^{m^R,n^G}$, and $A^{m^G,n^G}$ (where $A^{m^G,n^G}_{\text{kering}} = \emptyset$).

### 5.2.2 Parameters and Variables

The scheduling constraints rely on the following bounds and parameters:

- $d^+_{\textbf{max}}/d^-_{\textbf{max}}$: Maximum allowed delay or advance (set to 15 minutes).

- $l_{i,j}/u_{i,j} \in [0, T-1]$: Lower and upper bounds for activity duration, alongside specific minimum frontal headways $l^{\text{frontaal}}$.

- $standaard\_keertijd_{i,j}$: Turnaround time in the baseline schedule (0 if undefined).

- $F_m/G_n$: The final segment of train $m$ and the initial segment of train $n$, respectively.

- $dur_{m,f}$: Total duration of a given segment.

To resolve changes dynamically, the model uses the following deviation and infrastructure tracking variables:

- $d^+_{i,m}/d^-_{i,m}$: Non-negative delay and advance metrics relative to the baseline schedule $\pi_i$.

- $k_{i,j} \in \{0, 1\}$: Binary decision indicating whether event $i$ turns around onto event $j$.

- $keertijd_{i,j}$: Net change in turnaround time relative to $standaard\_keertijd_{i,j}$.

- $keertijd^+_{i,j}$: Turnaround duration exceeding the threshold value $gk$.

### 5.2.3 Objective Function

The objective function minimizes changes to the baseline timetable by penalizing schedule adjustments. It consists of three core components: discouraging train cancellations, limiting time deviations (delays and advances), and managing train turnaround choices. Different penalty weights ($w$) are assigned to balance these priorities, with passenger and freight services penalized separately. The complete objective function is formulated across the following sub-components:

$$\min \quad \sum_{m \in M} \sum_{f} w_{m,f}^{\text{cancel}} dur_{m,f} y_{m,f} + \sum_{m \in M} w_{m}^{\text{restart}} \gamma_m \tag{5.1}$$

$$+ \sum_{m \in M^R} \sum_{i \in E_m^f} w_i^{\text{delay,R}} d_{i,m}^{+} \tag{5.2}$$

$$+ \sum_{m \in M^R} \sum_{i \in E_m^f} w_i^{\text{advance,R}} d_{i,m}^{-} \tag{5.3}$$

$$+ \sum_{m \in M^G} \sum_{i \in E_m^f} w_i^{\text{delay,G}} d_{i,m}^{+} + \sum_{m \in M^G} \sum_{i \in E_m^f} w_i^{\text{advance,G}} d_{i,m}^{-} \tag{5.4}$$

$$+ \sum_{i,j \in A_{\text{reizigers}} \cup A_{\text{mixed}}} w_{i,j}^{\text{keertijd}} keertijd_{i,j} \tag{5.5}$$

$$+ \sum_{i,j \in A} w_{i,j}^{\text{keertijd}^+} keertijd_{i,j}^{+} \tag{5.6}$$

$$+ \sum_{i,j \in A_{\text{reizigers}} \cap A_{\text{voorkeurskering}}} w_{i,j}^{\text{voorkeurskering}} k_{i,j} \tag{5.7}$$

$$+ \sum_{i,j \in (A_{\text{mixed}} \cup A_{\text{LM}}) \cap A_{\text{voorkeurskering}}} w_{i,j}^{\text{voorkeurskering leegmat}} k_{i,j} \tag{5.8}$$

$$+ \sum_{i,j \in A_{\text{dummy\_kering}}} w_{i,j}^{\text{dummy kering}} k_{i,j} \tag{5.9}$$

To understand the trade-offs within the optimization, each block targets a specific operational priority:

- **Equation** (5.1) **(Train Cancellations):** Penalizes the total duration of canceled train segments ($y_{m,f}$) and penalizes instances where a train must perform a partial route restart ($\gamma_m$). This ensures that the model keeps as many baseline services running as possible.
- **Equation** (5.2) **(Passenger Delays):** Quantifies and penalizes positive time deviations ($d_{i,m}^{+}$) for passenger lines ($M^R$), ensuring that trains do not arrive or depart later than their original baseline window.

- **Equation** (5.3) **(Passenger Advances):** Penalizes negative time deviations ($d^-_{i,m}$) for passenger lines, preventing trains from running earlier than scheduled, which heavily affects passenger connections.
- **Equation** (5.4) **(Freight Deviations):** Separately weighs and penalizes both delays and advances for freight networks ($M^G$), accounting for the fact that freight scheduling has different operational tolerances than passenger logistics.
- **Equation** (5.5) **(Standard Turnaround Duration):** Apply a base penalty to the physical time a train spends turning around ($keertijd_{i,j}$) on active passenger and mixed lines to keep track usage efficient.
- **Equation** (5.6) **(Excessive Turnover Penalty):** Heavily penalizes instances where a turnaround duration exceeds the standard $gk$ buffer threshold ($keertijd^+_{i,j}$), preventing local delays from compounding across rolling stock allocations.
- **Equations** (5.7) **and** (5.8) **(Preferred Turnarounds):** Manage the tracking choices ($k_{i,j}$) by incentivizing the model to select historically preferred platform turnaround pathways for standard passenger lines and empty material stock (*leeg materieel*) respectively.
- **Equation** (5.9) **(Dummy Turnarounds):** Penalizes the use of artificial fallback or dummy turnaround routes, forcing the optimization framework to favor realistic infrastructure paths whenever feasible.

### 5.2.4 Operational Durations and Running Limits

Train movements must comply with physical safety and rolling stock limits throughout the network. The model enforces minimum and maximum duration thresholds for running segments ($A^m_{\text{rijden}}$) and platform station stops ($A^m_{\text{kortestop}} \cup A^m_{\text{langestop}}$). These bounds ensure realistic driving times and allow sufficient dwell time for passenger boarding, becoming automatically deactivated if a train segment is cancelled ($y_{m,f} = 1$).

$$l_{i,j}(1 - y_{m,f}) \leq v_j - v_i + Tq_{i,j} \leq u_{i,j}(1 - y_{m,f}) \quad \forall (i,j) \in A^m_{\text{rijden}},\ \forall m \in M \tag{5.10}$$

$$l_{i,j}(1 - y_{m,f} - y_{m,g}) \leq v_j - v_i + Tq_{i,j} \leq u_{i,j}(1 - y_{m,f} - y_{m,g}) + T(y_{m,f} + y_{m,g}) \tag{5.11}$$

$$\forall (i,j) \in A^m_{\text{kortestop}} \cup A^m_{\text{langestop}},\ \forall m \in M$$

### 5.2.5 Safety Headways and Track Separation

To prevent collisions on shared corridors and single-track lines, the model enforces sequence-dependent headway margins and head-on spacing limits ($l^{\text{frontaal}}$) between consecutive train movements. While passenger train operations ($M^R$) adhere to tight safety spacing, main-line macro-headways involving freight services ($M^G$) can be relaxed. High-resolution sequencing for freight cars is absorbed locally by secondary yard classification tracking rules rather than overburdening the global network solver.

$$l_{i,j}(1 - y_{m,f} - y_{n,g}) \leq v_j - v_i + Tq_{i,j} \leq u_{i,j}(1 - y_{m,f} - y_{n,g}) + T(y_{m,f} + y_{n,g}) \quad (5.12)$$
$$\forall (i,j) \in A^{m^R,n^R}_{\text{headway}} \cup A^{m^R,n^G}_{\text{headway}}$$

$$l_{i,j}(1-y_{m,f}-y_{n,g}) \leq v_j-v_i+Tq_{i,j} \leq T-l^{\text{frontaal}}(1-y_{m,f}-y_{n,g}) \quad \forall (i,j) \in A^{m^R,n^R}_{\text{frontaal}} \cup A^{m^R,n^G}_{\text{frontaal}} \quad (5.13)$$

### 5.2.6 Timing Shifts, Delays, and Advances

To guarantee that the train schedule remains functional, the PESP model can make three different types of adjustments to the original timetable: time shifting, turnaround tracking, and mid-route line interventions. These adjustments are quantified using the absolute deviation variables for the delay ($d^+_{i,m}$) and advance ($d^-_{i,m}$) for the train $m$. To prevent the network schedule from becoming unrecognizable to passengers 5.14, these structural deviations are limited to a maximum of 15 minutes ($d^+_{\max}$, $d^-_{\max}$) relative to the baseline timeline. In addition, the tracking parameter $\alpha_i$ accounts for shifts that wrap around cycle boundaries (such as midnight). Furthermore, if a segment is canceled using the intervention variable mid-route ($y_{m,f}$), its event times are forced to zero, dynamically rendering its driving, station stop, and vehicle safety constraints inactive 5.17.

$$0 \leq v_i \leq (T-1)(1 - y_{m,f}) \quad \forall i \in E^f_m, \forall f \in Seg_m, \forall m \in M \quad (5.14)$$

$$d^+_{i,m} \geq v_i + T \cdot \alpha_i - \pi_i(1 - y_{m,f}) \quad \forall i \in E^f_m, \forall f \in Seg_m, \forall m \in M \quad (5.15)$$

$$0 \leq d^+_{i,m} \leq \min(d^+_{\max}, \left\lfloor \frac{T}{2} \right\rfloor - 1), \quad \forall i \in E^f_m \, \forall f \in Seg_m, \forall m \in M \quad (5.16)$$

$$d^-_{i,m} \geq \pi_i(1 - y_{m,f}) - T \cdot \alpha_i - v_i \quad \forall i \in E^f_m, \forall f \in Seg_m, \forall m \in M \quad (5.17)$$

$$0 \leq d^-_{i,m} \leq \min(d^-_{\max}, \left\lfloor \frac{T}{2} \right\rfloor - 1), \forall i \in E^f_m \, \forall f \in Seg_m, \forall m \in M \quad (5.18)$$

### 5.2.7 Mid-Route Line Interventions and Continuity

When a maintenance blockade forces a line intervention, the model applies a series of continuity constraints to ensure that fractured train routes logically terminate or restart across adjacent segments (10). Constraint (5.19) prevents isolated cancellations by dictating that a segment can only be cancelled if the subsequent segment is either also cancelled or successfully restarted. Conversely, Constraints (5.20) and (5.21) govern the restart mechanism, establishing that a segment can only restart ($z_{m,f+1} = 1$) if its immediate predecessor was cancelled ($y_{m,f} = 1$) and the current segment itself remains active ($y_{m,f+1} = 0$). Finally, Constraint (5.22) accumulates these interventions into $\gamma_m$ to track the exact number of times a train movement starts or restarts along its journey, accounting for both its initial departure at the first segment ($G_m$) and any subsequent mid-route restarts.

$$y_{m,f} \leq z_{m,f+1} + y_{m,f+1} \quad \forall f \in Seg_m,\ f \neq F_m,\ \forall m \in M \tag{5.19}$$

$$z_{m,f+1} \leq y_{m,f} \quad \forall f \in Seg_m,\ f \neq F_m,\ \forall m \in M \tag{5.20}$$

$$z_{m,f+1} \leq 1 - y_{m,f+1} \quad \forall f \in Seg_m,\ f \neq F_m,\ \forall m \in M \tag{5.21}$$

$$\gamma_m \geq \sum_{f \neq G_m} z_{m,f} + (1 - y_{m,G_m}) \quad \forall m \in M \tag{5.22}$$

### 5.2.8 Turnaround Logic and Tracking

When maintenance disruptions force a line intervention, the model implements turnaround constraints to dynamically adjust rolling stock allocations via $k_{i,j}$ (21). To capture exactly where a route terminates or restarts, the framework establishes the helper states $y^*$ and $\widetilde{y}$. Constraints (5.24) and (5.25) use these states to enforce a mathematical lower bound on turnarounds: if a segment remains operational but its immediate successor is cancelled, an outgoing turnaround is strictly required ($\sum_j k_{i,j} \geq 1$); conversely, an incoming turnaround is triggered when a train route safely restarts following a cancelled segment.

Upper bounds are governed by Constraints (5.26) through (5.29), ensuring turnarounds only occur on active segments and only when adjacent segments are structurally disrupted. These valid turnaround pairs are restricted to predefined paths; for instance, empty stock movements (*leeg materieel*) are constrained to fixed fallback tracks, meaning they are automatically cancelled if their specific feed-in service is lost. Finally, Constraint (5.30) tracks operational buffer quality: if an adjusted turnaround duration ($keertijd_{i,j}$) exceeds the standard threshold value $gk$, the surplus is flagged via the non-negative penalty variable

$keertijd^{+}_{i,j}$ to prevent crew and rolling stock delays from compounding across the network (7, 33).

$$y^* = \begin{cases} y_{m,f+1} & \text{when } f+1 \text{ exists for } m \in M^R, \\ 1 & \text{otherwise.} \end{cases}$$
$$\widetilde{y} = \begin{cases} y_{n,g-1} & \text{when } g-1 \text{ exists for } n \in M^R, \\ 1 & \text{otherwise.} \end{cases} \tag{5.23}$$

$$\sum_j k_{i,j} \geq y^* - y_{m,f} \quad \forall i \in E^f_m,\ \forall f \in Seg_m,\ \forall m \in M^R \tag{5.24}$$

$$\sum_i k_{i,j} \geq \widetilde{y} - y_{n,g} \quad \forall j \in E^g_n,\ Bowen \forall g \in Seg_n,\ \forall n \in M^R \tag{5.25}$$

$$y_{m,f} + \sum_j k_{i,j} \leq 1 \quad \forall i \in E^f_m,\ \forall f \in Seg_m,\ \forall m \in M^R \tag{5.26}$$

$$\sum_j k_{i,j} \leq y_{m,f+1} \quad \forall i \in E^f_m,\ \forall f \in Seg_m,\ f \neq F_m,\ \forall m \in M^R \tag{5.27}$$

$$y_{n,g} + \sum_i k_{i,j} \leq 1 \quad \forall j \in E^g_n,\ \forall g \in Seg_n,\ \forall n \in M^R \tag{5.28}$$

$$\sum_i k_{i,j} \leq y_{n,g-1} \quad \forall j \in E^g_n,\ \forall g \in Seg_n,\ g \neq G_n,\ \forall n \in M^R \tag{5.29}$$

$$keertijd^{+}_{i,j} \geq keertijd_{i,j} - gk \quad \forall (i,j) \in A \tag{5.30}$$

### 5.2.9 Border Timing Constraints

International border events must comply with strict timing windows dictated by international agreements. While border crossing times generally remain fixed to the baseline schedule, a train may adaptively inherit a border time slot from another train running on the same border corridor and direction. For freight trains ($M^G$), slots can be shared between movements with operational validities that match. For passenger classes (such as Sprinters, Intercities, Nightjets, and International services), slots can be shared with other movements from the same train series, allowing a predefined tolerance window (± minutes) relative to the original schedule.

To model this choice, the binary decision parameter $g_{i,k} \in \{0, 1\}$ indicates whether border event $i \in E_{\text{border}}$ is scheduled within a specific predefined border interval $k$, bounded by $[l_{i,k}^{\text{border}}, u_{i,k}^{\text{border}}]$.

$$v_i \geq l_{i,k}^{\text{border}} - l_{i,k}^{\text{border}}(1 - g_{i,k}) \quad \forall i \in E_{\text{border}},\ \forall k \tag{5.31}$$

$$v_i \leq u_{i,k}^{\text{border}} + (T - u_{i,k}^{\text{border}})(1 - g_{i,k}) \quad \forall i \in E_{\text{border}},\ \forall k \tag{5.32}$$

$$\sum_k g_{i,k} = 1 - y_{m,f} \quad \forall i \in E_{\text{border}} \text{ with segment } f \text{ of train } m \tag{5.33}$$

Constraints (5.31) and (5.32) enforce that the adjusted event time $v_i$ respect the lower and upper bounds of the chosen time window when $g_{i,k} = 1$. If a different interval is selected, the terms scale appropriately to make the boundaries redundant. Finally, Constraint (5.33) ensures that exactly one valid border interval is selected, unless the underlying train segment is cancelled ($y_{m,f} = 1$), which forces all associated $g_{i,k}$ indicators to zero and dynamically deactivates the boundary restrictions.

### 5.2.10 The Sliding Window Solution Methodology

The model uses a rolling window protocal to make sure that it is scalable across broad networks. It breaks up the time interval into blocks and incrementally slides over the full window. For each sliding movement, a section overlaps with the previous window while the remaining part optimizes the new set by dividing the timeline into smaller solve windows (`INTERVALS_TO_SOLVE`) and fix intervals (`INTERVALS_TO_FIX`). Within this sequential loop, the model first identifies all train movements that cross the temporal boundaries or link to immediate turnarounds. Before sliding to the next window, it freezes the macro event parameters ($v$), the baseline turnaround variables ($k$) and the segment statuses ($y$) falling within the designated fix interval, which protects the newly computed part of the schedule. To accelerate the process, each new segment is given a warm start via the solution of the previous window. Finally, the solver disables all non-relevant track configurations outside the active interval by forcing structural cancelations, concentrating computational resources entirely on executing the targeted window.

## 5.3 The Microscopic Layer: Station Capacity Model

The Station Capacity Model (SCM) acts as a micro layer that verifies the capacity and the interlocking line routing of a *Dienstregelpunt* (DRP) (15, 25). It checks specific station capacity limitations for DRP subsets where the macro layer proposes schedule variations (25).

### 5.3.1 Sets and Indices

The station infrastructure is organized into sets. The main set $S$ includes all track lines, divided into single tracks ($S^e$) and double tracks ($S^d$), where $E^s \subseteq S^e$ denotes single tracks forming double tracks, and $D^s \subseteq S^d$ identifies double tracks containing single tracks. Train traffic is managed by set $M$, which defines valid track occupations ($S^m \subseteq S$). Operational aspects are captured in $B$, the set of train series with directional movements, linked to specific occupations ($M^b \subseteq M$). Safety is ensured by sets identifying conflicts, such as overlapping passenger ($Y$) and freight ($Y^g$) occupancy, as well as sequential platform arrival sets ($R$ and $R^g$). Critical track interactions are monitored through sets $K$ and $K^g$ for crossover spacing, and $L$ and $L^g$ for platform conflict detection.

### 5.3.2 Parameters

The baseline station routing choices and platform pattern consistency rely on a set of general structural parameters, variables, and matching allocation bounds:

- $A_{m,s} \in \{0, 1\}$: Accessibility parameter; 1 if train occupation $m$ physically fits and can access track $s$, 0 otherwise.
- $P_{m,s} \in \{0, 1\}$: Baseline preference parameter; 1 if train occupation $m$ is assigned to track $s$ in the original timetable, 0 otherwise.
- $X_{m,s} \in \{0, 1\}$: Routing decision variable; 1 if train occupation $m$ is assigned to track $s$, 0 otherwise.
- $F_{b,s} \in \mathbb{R}$: Variable tracking the total number of train occupations from series $b$ allocated to station track $s$.
- $Z_b \in \mathbb{R}$: Variable tracking the maximum number of train occupations within series $b$ sharing a single common track.

- $G_{m,s} \in \{0,1\}$: Penalty variable active if occupation $m$ does not use the primary track of its corresponding train series.
- $H_b \in \{0,1\}$: Bonus variable active if more than half of the train services within series $b$ share a single track asset.

### 5.3.3 Objective Function

The objective function minimizes the occurrence and total duration of all microscopic conflicts. Secondary goals include maintaining original baseline tracks and maximizing the consistency of the platform pattern by grouping identical train series $b$ into identical track assets using penalty and bonus weights ($w$). The complete microscopic objective function is formulated across the following sub-components:

$$\min \quad \sum_{m,n} \sum_{s \in S^m \cup S^n} w^{\text{spoorbezettingconflict}} \text{conflict}^{\text{spoorbezetting}}_{m,n,s} \tag{5.34}$$

$$+ \sum_{m,n} \sum_{s \in S^m \cup S^n} w^{\text{duur\_spoorbezettingconflict}} \text{conflict}^{\text{spoorbezetting}}_{m,n,s} \cdot d^{\text{spoorbezettingconflict}}_{m,n} \tag{5.35}$$

$$+ \sum_{m,n} \sum_{s \in S^m \cup S^n} w^{\text{perronnorm}} \text{conflict}^{\text{perronnorm}}_{m,n,s} \tag{5.36}$$

$$+ \sum_{m,n} \sum_{s \in S^m \cup S^n} w^{\text{duur\_perronnormconflict}} \text{conflict}^{\text{perronnorm}}_{m,n,s} \cdot d^{\text{perronnorm}}_{m,n} \tag{5.37}$$

$$+ \sum_{m,n} \sum_{s \in S^m, s' \in S^n} w^{\text{overkruisconflict}} \text{conflict}^{\text{overkruis}}_{m,n,s,s'} \tag{5.38}$$

$$+ \sum_{m,n} \sum_{s \in S^m, s' \in S^n} w^{\text{duur\_overkruisconflict}} \text{conflict}^{\text{overkruis}}_{m,n,s,s'} \cdot d^{\text{overkruis}}_{m,n,s,s'} \tag{5.39}$$

$$- \sum_{m} \sum_{s} w^{\text{zelfde spoor}} P_{m,s} X_{m,s} \tag{5.40}$$

$$+ \sum_{m} \sum_{s \in S^m} w^{\text{serie\_richting}} G_{m,s} \tag{5.41}$$

$$+ \sum_{b} w^{\text{majority\_serie\_richting}} H_b \tag{5.42}$$

To understand the operational trade-offs inside this station-level optimization, each block targets a specific microscopic safety or quality metric:

- **Equation** (5.34) **(Track Occupation Conflicts):** Penalizes the absolute occurrence of simultaneous track occupation conflicts ($\text{conflict}^{\text{spoorbezetting}}_{m,n,s}$) where two trains try to claim the exact same physical infrastructure element at station $s$.

- **Equation** (5.35) **(Track Conflict Duration):** Penalizes the exact time duration ($d_{m,n}^{\text{spoorbezettingconflict}}$) of those overlapping track occupation events to ensure unavoidable conflicts are kept as brief as possible.

- **Equation** (5.36) **(Platform Norm Conflicts):** Penalizes violations of specific platform safety and spacing regulations ($\text{conflict}_{m,n,s}^{\text{perronnorm}}$), protecting safe headway buffers between successive arrivals and departures at passenger platforms.

- **Equation** (5.37) **(Platform Norm Conflict Duration):** Accounts for and minimizes the temporal length ($d_{m,n}^{\text{perronnorm}}$) of any platform norm violations, reducing platform congestion risks.

- **Equation** (5.38) **(Route Crossing Conflicts):** Tracks and penalizes instances where paths cross ($\text{conflict}_{m,n,s,s'}^{\text{overkruis}}$), occurring when a train's arrival or departure route intersects and blocks the track trajectory of an opposing train.

- **Equation** (5.39) **(Crossing Conflict Duration):** Minimizes the duration ($d_{m,n,s,s'}^{\text{overkruis}}$) of crossing interlocking overlaps to keep switches clear and prevent gridlocks at busy switch areas.

- **Equation** (5.40) **(Original Baseline Reward):** Features a negative sign, acting as a bonus reward for routing train $m$ along its original preferred track $s$ ($X_{m,s}$) according to reference profile indicators ($P_{m,s}$), ensuring deviations from the baseline remain small.

- **Equation** (5.41) **(Series Track Deviations):** Penalizes instances ($G_{m,s}$) where an individual train must deviate from its standard series-wide track assignment rules, protecting standard station routing conventions.

- **Equation** (5.42) **(Series Pattern Consistency):** Penalizes a lack of platform pattern uniformity across a whole train series $b$ ($H_b$). This forces the algorithm to prioritize recognizable, consistent platform patterns for passengers (e.g., ensuring all Intercity trains of a specific line use the same platform group).

### 5.3.4 Constraints

Every train occupation must be mapped to a single physical track element matching its dimensions. Track choices must adhere to infrastructure accessibility parameters, and

selecting a double track automatically blocks its single sub-components:

$$\sum_{s' \in S^e} X_{m,s'} = 1 + \sum_{s \in S^d} X_{m,s} \quad \forall m \in M \tag{5.43}$$

$$\sum_{s \in S^d} X_{m,s} \leq 1 \quad \forall m \in M \tag{5.44}$$

$$X_{m,s} \leq X_{m,s'} \quad \forall m \in M, \forall s \in S^d, s' \in E^s \tag{5.45}$$

$$X_{m,s} \leq A_{m,s} + \sum_{s' \in D^s} X_{m,s'} \quad \forall m \in M, \forall s \in S \tag{5.46}$$

To streamline passenger operations, tracking structures group train services sharing an identical line series ($b \in B$) onto the same physical platforms to establish regular-interval patterns:

$$F_{b,s} = \sum_{m \in M^b \cap M^s} X_{m,s} \quad \forall b \in B, \forall s \in S \tag{5.47}$$

$$Z_b \geq F_{b,s} \quad \forall b \in B, \forall s \in S \tag{5.48}$$

$$F_{b,s} \geq Z_b - (G_{m,s} + 1 - X_{m,s}) \cdot |M^b| \quad \forall b \in B, \forall m \in M^b, \forall s \in S^m \tag{5.49}$$

$$G_{m,s} \geq X_{m,s} \quad \forall m \in M, \forall s \in S^m \tag{5.50}$$

$$H_b \leq \frac{2 \cdot (Z_b - 1)}{|M^b|} \quad \forall b \in B \tag{5.51}$$

$$X_{m,s} + \sum_{n|(m,n) \in B} \sum_{s' \in S^n, s' \neq s} X_{n,s'} \leq 1 + F_{m,s} \qquad \forall m \in M, \quad \forall s \in S^m \tag{5.52}$$

Constraints (5.47) through (5.52) regulate platform pattern uniformity. They apply penalty and bonus states to prioritize uniform scheduling over split options.

### 5.3.5 Microscopic SCM Conflict Definitions

The SCM checks for conflicts using four specific microscopic safety spacing rules evaluated over overlapping train configurations.

#### Track Occupancy (*Spoorbezetting*) Conflicts

An occupancy conflict indicates that two separate passenger trains are assigned to the same platform section during overlapping time intervals ($Y \setminus Y^g$):

- $d_{m,n}^{\text{spoorbezettingconflict}}$: The time duration of an overlapping track occupancy conflict, including the platform norm cushion.

- $\text{conflict}_{m,n,s}^{\text{spoorbezetting}} \in \{0,1\}$: evaluates to 1 if an overlapping track occupancy conflict occurs on track $s$, 0 otherwise.

$$X_{m,s} + X_{n,s} \leq 1 + \text{conflict}_{m,n,s}^{\text{spoorbezetting}} \quad \forall (m,n) \in Y \setminus Y^g,\, \forall s \in S^m \cup S^n \tag{5.53}$$

Constraint (5.53) forces the tracking flag to 1 if the arrival and departure profiles of active passenger lines overlap simultaneously on an identical asset $s$.

### Platform Follow-Up (*Perronnorm*) Conflicts

Trains using the same platform track following directly after each other must be separated by safety buffers based on their operational event types ($R \setminus R^g$):

- $d_{m,n}^{\text{perronnormconflict}}$: The time duration of an overlapping platform follow-up sequence conflict.

- $\text{conflict}_{m,n,s}^{\text{perronnorm}} \in \{0,1\}$: evaluates to 1 if a platform follow-up sequence conflict occurs on track $s$, 0 otherwise.

**Table 5.1:** Platform Follow-Up Safety Cushion Parameters (Minutes)

| Preceding / Following Event | Arrival (A) | Pass (D) | Short Stop (K) | Departure (V) |
|---|---|---|---|---|
| **Arrival (A)** | 3 | 5 | 5 | 1 |
| **Pass (D)** | 3 | 3 | 3 | 4 |
| **Short Stop (K)** | 4 | 4 | 4 | 5 |
| **Departure (V)** | 4 | 4 | 4 | 2 |

$$X_{m,s} + X_{n,s} \leq 1 + \text{conflict}_{m,n,s}^{\text{perronnorm}} \quad \forall (m,n) \in R \setminus R^g,\, \forall s \in S^m \cup S^n \tag{5.54}$$

If a sequential platform block utilization violates the minimum required headway cushions outlined in Table 5.1, a platform follow-up conflict is flagged via relation (5.54).

### Track Crossover (*Overkruis*) Conflicts

A crossover conflict captures overlapping paths across shared switches or layout throat junctions where crossing margins are violated ($K \setminus K^g$):

- $d_{m,n,s,s'}^{\text{overkruisconflict}}$: The time duration of an overlapping throat crossover intersection conflict between track allocations $s$ and $s'$.

- $\text{conflict}_{m,n,s,s'}^{\text{overkruis}} \in \{0,1\}$: evaluates to 1 if a crossover intersection conflict occurs between track allocations $s$ and $s'$, 0 otherwise.

**Table 5.2:** Intersection Safety Buffers — Same Direction (Minutes)

| Preceding / Following Event | Arrival (A) | Pass (D) | Short Stop (K) | Departure (V) |
|---|---|---|---|---|
| **Arrival (A)** | 3 | 2 | 3 | 1 |
| **Pass (D)** | 3 | 3 | 3 | 2 |
| **Short Stop (K)** | 3 | 3 | 3 | 2 |
| **Departure (V)** | 4 | 3 | 3 | 2 |

**Table 5.3:** Intersection Safety Buffers — Opposing Direction (Minutes)

| Preceding / Following Event | Arrival (A) | Pass (D) | Short Stop (K) | Departure (V) |
|---|---|---|---|---|
| **Arrival (A)** | 3 | 2 | 1 | 1 |
| **Pass (D)** | 4 | 3 | 4 | 1 |
| **Short Stop (K)** | 6 | 5 | 6 | 1 |
| **Departure (V)** | 6 | 5 | 6 | 2 |

$$X_{m,s} + X_{n,s'} \leq 1 + \text{conflict}^{\text{overkruis}}_{m,n,s,s'} \quad \forall (m,n,s,s') \in K \setminus K^g \tag{5.55}$$

Throat crossing path overlaps are identified and penalized through Constraint (5.55) based on the directional orientation buffers specified in Tables 5.2 and 5.3.

### Route-Through-Platform (*Rijweg door spoorbezetting*) Conflicts

This conflict occurs when the entrance or exit path of a train physically cuts through a platform block currently occupied by another stationary vehicle ($L \setminus L^g$):

- $d^{\text{rijweg door spoorbezetting}}_{m,n,s,s'}$: The time duration of an overlapping route-through-platform conflict between track allocations $s$ and $s'$.
- $\text{conflict}^{\text{rijweg door spoorbezetting}}_{m,n,s,s'} \in \{0,1\}$: evaluates to 1 if a route-through-platform path conflict occurs between track allocations $s$ and $s'$, 0 otherwise.

$$X_{m,s} + X_{n,s'} \leq 1 + \text{conflict}^{\text{rijweg door spoorbezetting}}_{m,n,s,s'} \quad \forall (m,n,s,s') \in L \setminus L^g \tag{5.56}$$

Path separation is enforced via Constraint (5.56) by matching interlocking paths against platform occupancy times.

Freight train paths do not trigger occupancy or follow-up conflicts ($Y^g, R^g, K^g, L^g = \emptyset$), as railway allocation guidelines allow overlapping freight paths during macro-planning phases. Because freight services lack explicit turnarounds, a standard 20-minute dwell calculation is enforced to ensure vehicles clear layout sections and release station capacity safely.

## 5.4 The Macro-Micro Alternating Iterative Loop

Some DRPs cannot be solved to a conflict-free state by the SCM algorithm, these DRPs give back a set of constraints to make sure the micro layer can become conflict-free (15). To reduce computation time, a decentralized search mechanism excludes nodes that have not changed, when the macro input variables of a station are identical to those of the previous iteration, its local feasibility is ensured to be unchanged and its SCM evaluation is automatically skipped.(15). When the SCM detects a station conflict, specialized linear capacity constraints are generated and appended to the macro PESP model for its next run (15, 27). These feedback cuts are restricted to the subset of blocked stations ($S^{sc}$).

### 5.4.1 Station Track and Platform Capacity Feedback Cuts

Let $c_{i,j} \in \{0,1\}$ define a binary indicator set to 1 if event $i$ holds a platform track when event $j$ arrives. The framework forces $c_{i,j}$ to 1 if the time interval falls below the required safety buffer $\delta_{i,j}$ (10):

$$\delta_{i,j}\,(1 - c_{i,j} - y_{m,f} - y_{n,g}) \le v_j - v_i + T \cdot q_{i,j} \le (T - \epsilon) - \delta_{j,i}\,(1 - c_{ji} - y_{m,f} - y_{n,g}) \quad (5.57)$$

$$\forall (i,j) \in A_{\text{sc}}^{m,n}(s),\ \forall m,n \in M,\ \forall s \in S^{sc}$$

The model aggregates these elements to ensure that the global count of active occupations fits within the station's platform tracks $P(s)$ and through-running lines $D(s)$ (25):

$$\sum_{i \in O_j^-(s)} c_{i,j} \le P(s) - 1 \quad \forall j \in E(s),\ \forall s \in S^{sc} \quad (5.58)$$

$$\sum_{i \in O_j^+(s)} c_{i,j} \le P(s) + D(s) - 1 \quad \forall j \in E(s),\ \forall s \in S^{sc} \quad (5.59)$$

To align with freight yard properties, freight variables are synchronized ($c_{i,j} = c_{i',j}$) so that overlapping freight lines count as a single resource slot (3):

$$c_{i,j} = c_{i',j} \quad \forall i,i' \in E^g(s),\ i \ne i',\ \forall j \in E(s) \setminus E^g(s),\ \forall s \in S^{sc} \quad (5.60)$$

### 5.4.2 Station Intersection (Overkruis) Capacity Feedback Cuts

For track switches where routes intersect, path overlaps are bounded within a dedicated conflict set $S^{\text{sc\_overkruis}}$ (10). A binary indicator $c_{i,j}^{\text{overkruis}}$ marks a path conflict occurring within an intersection safety window $\delta_{i,j}^{\text{overkruis}}$ (22):

$$\delta_{i,j}^{\text{overkruis}}\left(1 - c_{i,j}^{\text{overkruis}} - y_{m,f} - y_{n,g}\right) \le v_j - v_i + T \cdot q_{i,j} \le T - \epsilon \quad (5.61)$$

$$\forall (i,j) \in A^{m,n}_{\text{sc\_overkruis}}(s),\ \forall m,n \in M,\ \forall s \in S^{\text{sc\_overkruis}}$$
$$0 \leq v_i - v_j + Tq_{j,i} \leq T - \epsilon - \delta^{\text{overkruis}}_{i,j}\left(1 - c^{\text{overkruis}}_{i,j} - y_{m,f} - y_{n,g}\right) \tag{5.62}$$
$$\forall (i,j) \in A^{m,n}_{\text{sc\_overkruis}}(s),\ \forall m,n \in M,\ \forall s \in S^{\text{sc\_overkruis}}$$

The intersection variables are aggregated to ensure throat movements conform to channel tracking capacities ($\gamma$):

$$\sum_{i \in OK^{+}_{j,\gamma}(s)} c^{\text{overkruis}}_{i,j} \leq P_{j,\gamma}(s) \quad \forall j \in E^{\text{overkruis}}(s),\ \forall \gamma \in \Gamma_j(s),\ \forall s \in S^{\text{sc\_overkruis}} \tag{5.63}$$

$$\sum_{i \in OK^{+}_{j,\gamma}(s)} c^{\text{overkruis}}_{i,j} \leq P_{j,\gamma}(s) + D_{j,\gamma}(s) \quad \forall j \in E^{\text{overkruis}}(s),\ \forall \gamma \in \Gamma_j(s),\ \forall s \in S^{\text{sc\_overkruis}} \tag{5.64}$$

## 5.5 The Bezem Iteration

The full model performs several iterations, but the last iteration is a bit different from the others. Instead of giving back conflicts to use as constraints, the approach is to make the micro layer conflict free by canceling the train movements that still give conflicts. This forces the model to converge and results in a feasible solution on both levels. This iteration is called the **Bezem Iteration**

1. **Targeted Cancellations ($M^{\text{cancel}}$):** A greedy tracking algorithm counts train occurrences across unresolved platform conflicts and selects the most critical paths to be canceled to maximize the remaining service level (34). For remaining switch intersection conflicts, the system automatically cancels the first train in the conflict pair to break deadlocks (35). These properties are enforced via structural cancellation rules:

$$y_s = 1 \quad \forall s \in S_m,\ \forall m \in M^{\text{cancel}} \quad \text{and} \quad y_s = 1 \quad \forall s \in S^{\text{cancel,prev}} \tag{5.65}$$

2. **Fixing Plan Deviations:** Delay ($d^+$) and advance ($d^-$) adjustments for all remaining, non-canceled train events ($E^{\text{allowed}}$) are locked to their previous values to preserve the timetable layout (34):

$$d^+_i = \delta^{+,\text{prev}}_i \quad \forall i \in E^{\text{allowed}} \tag{5.66}$$

$$d^-_i = \delta^{-,\text{prev}}_i \quad \forall i \in E^{\text{allowed}} \tag{5.67}$$

3. **Locking Turnarounds ($A^{\mathbf{fix}}$):** Essential platform turnarounds required by the remaining services are frozen to maintain rolling stock and crew continuity across the network layout (33, 35):

$$k_a = 1 \quad \forall a \in A^{\text{fix}} \tag{5.68}$$

This final optimization pass eliminates outstanding station layout conflicts, outputting an operationally deterministic, stable timetable (34).

# 6

# Data

This thesis examines train operations and railway infrastructure, relying on three main data types:

- **Timetable Data**: Scheduled departure/arrival times, station locations, routing, and track allocations.
- **Infrastructure Data**: Network layout, block sections, switches, track capacities, and operational constraints.
- **Historical Availability Data**: Records of planned and unplanned track closures used to simulate reduced-capacity scenarios.

All required data is provided by ProRail and stored securely. Since it does not contain direct passenger information, it has low sensitivity. The data set covers the entire Dutch railway system and includes current and planned schedules, representing a macroscopic network model.

## 6.1 Characteristics of the Data

Although the exact scale of the data changes depending on the subset or the localized corridor extracted for simulation, the global dataset encompasses the following metrics.

- Events in the schedule that include departures, arrivals, and turns.
- Material-specific information about the trains, detailing both the number and the specific types of wagon used.

- Multivariate attributes per train, including exact timing windows, track allocations, route paths, and dwell times.
- A detailed infrastructure graph consisting of nodes (representing stations, signals, and switches) and edges (representing tracks and block sections).
- Track closure records complete with timestamps, affected sections, and durations of disruption of 2027.

### 6.1.1 Data Integration and the ETL Pipeline

To feed the macro-micro scheduling model, data must be integrated from various distinct corporate and operational systems across ProRail and the main passenger operator, NS. These source datasets include *InfraAtlas* (the physical Dutch railway topology and planned 2027 maintenance windows, or *Buitendienststellingen*), the *Corridorboek* (international and freight detour routing matrices for 2025–2027), *Lookups* (rolling stock attributes and location constraints for 2025–2027), *Donna BVI* (daily/hourly infrastructure schedules and switch crossover movements for 2025–2027), *LAIKA* (an application supporting the European Time Table Redesign initiative for early international scheduling), and the *Dienstregeling* module, which merges Donna and LAIKA to create a unified railway schedule.

To process the different types of information, a standardized system called Extract, Transform, and Load (ETL) is used. The Python library Pandera helps to maintain high data quality in two steps. First, the raw incoming data is checked to ensure it fits the source standards. Then, the transformed data is verified with output standards to fix errors, remove null values, and standardize data types. After this verification, specific algorithms are applied within the ETL system to prepare the data for modeling.

- **Network Topology & Detours:** InfraAtlas data is stripped of duplicate platforms, compared to Donna to insert missing routes, and augmented with switch connections to build a complete baseline network graph alongside an adjusted graph reflecting active maintenance zones. To find valid detours in the reduced network, the model executes a multi-source Dijkstra algorithm, which "uses classical shortest-path routing by simultaneously initializing search frontiers from a set of multiple origin nodes toward a target destination node, efficiently resolving valid detour paths through the surviving network topology" (36). Similarly, the Corridorboek data are filtered and bidirectionally duplicated to ensure robust rerouting for freight and international trains.

- **Operational Constraints & Crossovers:** The Lookups dataset is transformed to compute platform thresholds and turning times, since "the time required to reverse a train depends on the specifications of the vehicle and the layout of the platform" (24). Meanwhile, the Donna BVI module extracts infrastructure metadata and tracks the exact sequences of switches utilized per route. This is critical because "in busy railway systems, congestion often occurs in front of the stations rather than on open tracks, making it difficult to find conflict-free paths as train frequencies increase." This precise tracking allows the model to map potential crossover conflicts at switches before executing the scheduling layer.

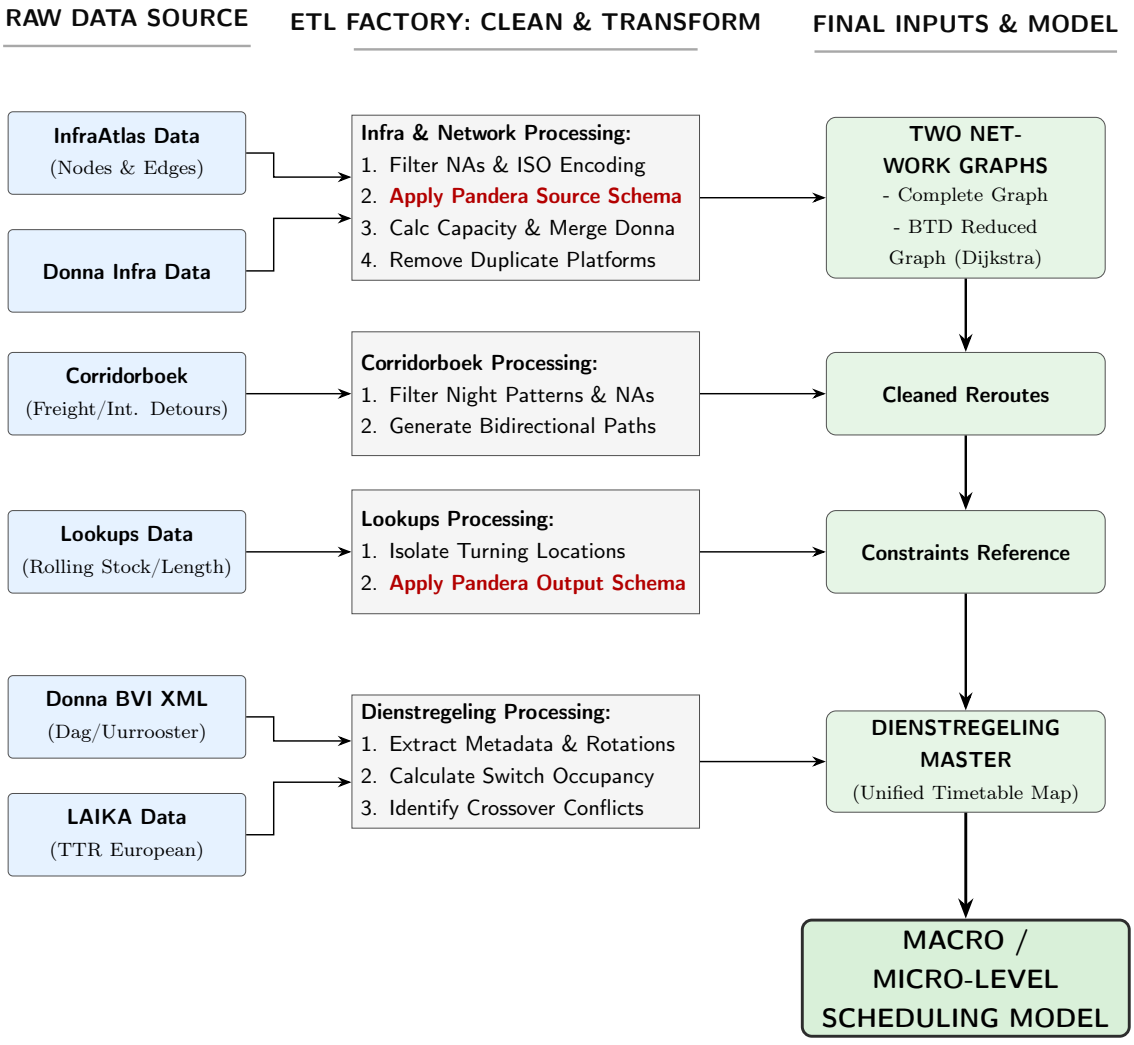


**Figure 6.1:** Data pipeline architecture and ETL processing workflow.

## 6.2 Data Exploration and Descriptive Analysis

In order to fully understand what parts of the model can be optimized and what the problem areas are, a data exploration is conducted. This section establishes a deep understanding of the underlying datasets. The baseline network contains 761 DRPs operating within a high-frequency rail network.

### 6.2.1 Simulation Framework and Data Structure

For this research, the data includes five maintenance test case scenarios: `dordrecht_lagezwaluwe`, `ede_arnhem`, `houtencastellum_geldermalsen`, `rotterdam_gouda_2027`, and `tilburg_vierkant_dicht`. To reduce the computational size of each test case, the full-day timetable is divided into eight three-hour windows: $(0, 3)$, $(3, 6)$, $(6, 9)$, $(9, 12)$, $(12, 15)$, $(15, 18)$, $(18, 21)$, and $(21, 24)$. The model then solves the scheduling problem for each specific time block individually. This creates 8 variations of each maintenance scenario. To evaluate each scenario, a three-iteration sequence is used. Each of the first two iterations includes a macro run, a micro run on all changed DRPs, and a feedback loop where micro-level constraints are sent back to the macro model. The third iteration serves as a final sweep, where a final macro run is performed, and any leftover micro conflicts are resolved by canceling those specific trains. Because of these distinct subsets, the full dataset contains features of each DRP across 123 distinct scenarios, resulting in a total dataset of approximately 95,000 rows. Several features are extracted from these different runs, logging different variables of each active DRP per execution. As shown in Table 6.1, there are four different types of features: run identifiers, predetermined infrastructure, PESP modifications, and conflict-related features. The predetermined infrastructure characteristics, such as the number of platforms, stay the same for each run, but the PESP modifications and conflicts change. This is because the PESP model searches for an optimal timetable every run, often incorporating new constraints from the SCM. As a result, the trains within the DRPs are modified differently each time, creating different conflicts.

### 6.2.2 Infrastructure Capacity and Conflict Analysis

Timetable traffic data show that throughput train movements (*doorkomst*) rule at 72%, followed by short stops (*korte stop*) at 14%, and terminal arrivals/departures (*aankomst/vertrek*) at 13%, maintaining a tight average dwell time (*bezettingstijd*) of 1.325 minutes. However, volume peaks during rush hours, with a single DRP handling an average of 59 trains between 06:00 and 09:00.

**Table 6.1:** Operational Log Dataset Schema and Feature Parameter Classification

| Variable Name | Operational Feature Description | Parameter Type |
|---|---|---|
| `drp_name` | Distinct Routing Point code identifier. | Run identifier |
| `testcase` | Maintenance geographical baseline boundary identifier. | Run identifier |
| `timeperiod` | Scheduled three-hour operational clock window. | Run identifier |
| `timestamp` | Hexadecimal unique validation run execution marker. | Run identifier |
| `iteration` | Sequential run block layer identifier (1, 2, or 3). | Run identifier |
| `size` | The total number of physical track platforms at the DRP. | Predetermined infrastructure |
| `categorical_size` | Structural scale classification (`small`, `medium`, `large`). | Predetermined infrastructure |
| `number of trains` | Number of trains scheduled through the DRP. | Predetermined infrastructure |
| `BUSYNESS` | Track capacity utilization ratio versus idle time. | Predetermined infrastructure |
| `total_changes` | Sum total of all macroscopic modifications applied to the DRP. | PESP modifications |
| `tijdsverleggingen` | Total count of timeshifted train paths (*Time Shifts*). | PESP modifications |
| `keringen` | Total count of platform train turnarounds executed (*Turnarounds*). | PESP modifications |
| `omleidingen` | Total count of trains rerouted(*Reroutes*). | PESP modifications |
| `solved` / `solved_time` | SCM internal resolution state flag and elapsed solve time. | PESP modifications |
| `is_changed` | Binary variable indicating if DRP is changed by a PESP adjustment. | Conflict related |
| `has_conflict` | Binary variable indicating if the DRP has conflicts after SCM computation. | Conflict related |
| `possible_conflicts` | Number of overlapping track possessions that could create conflicts per DRP. | Conflict related |
| `ignorable_conflicts` | Number of conflicts that already occured in the original timetable per DRP. | Conflict related |
| `actual_conflicts` | Total number of conflicts identified after SCM computation per DRP. | Conflict related |

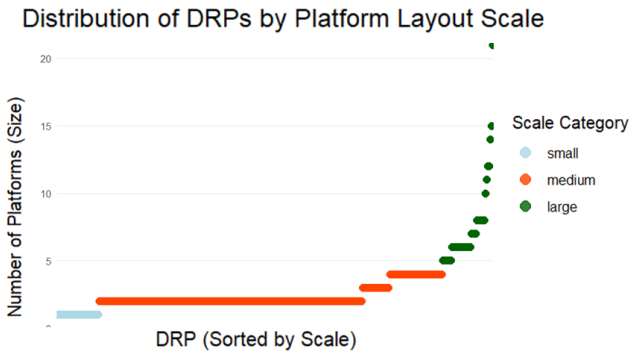


**Figure 6.2:** Distribution of DRPs Sorted by Platform Layout Scale.

The physical size of a station varies significantly, averaging 2.5 platforms across the network. To structure this variation, DRPs are categorized into three groups: *Small* (1 platform; 25%), *Medium* (2 to 5 platforms; 64%), and *Large* (above 5 platforms up to a maximum of 23; 11%), as mapped in Figure 6.2.

Filtering the data down to only nodes changed by PESP (is_changed = TRUE) reveals immense room for optimization. Out of these observations, the dataset records 4,118 verified conflict instances (has_conflict = TRUE) against 21,572 compliant instances. As illustrated in Figure 6.3, while a typical active run window forces the evaluation of 50 to 250 changed DRPs, the mean number of verified actual conflicts consistently stays below 50. Possible conflicts float roughly 5% to 15% beneath the total changed volume. Possible

conflicts are determined by verifying for all trains in the DRP whether they could conflict with each other by overlapping track possessions.

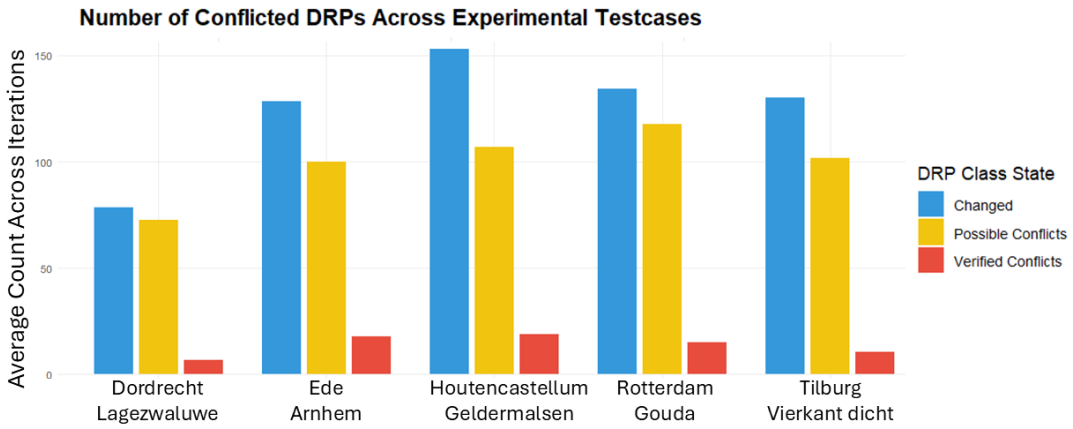


**Figure 6.3:** Mean Count of DRP States per Active Run.

Mapping these conflicts reveals a surprise with respect to the busyness ratio. Figure 6.4 compares the number of verified conflicts with the busyness of the station. The data points form a scattered cloud with no clear upward trend. Stations with very low traffic frequently experience severe conflict spikes, while busy stations often remain completely conflict-free. This visual independence shows that traffic density alone is a poor predictor of actual conflict risk. Instead, scheduling deadlocks are driven by more complex, hidden geometric bottlenecks than just a high number of trains.

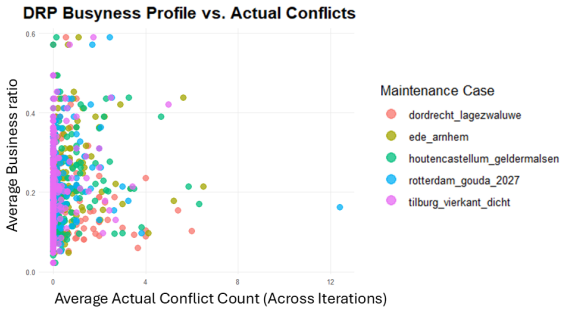


**Figure 6.4:** Capacity Utilization vs. Verified Conflicts.

### 6.2.3 Macroscopic PESP Schedule Interventions

PESP has the capability to make three types of changes: rolling-stock turnarounds (*keringen*), time shifts (*tijdsverleggingen*) and train rerouting (*omleidingen*). In Figure 6.5 the

division of these changes conducted by the PESP model is visualized in a pie chart. There is no clear preference for a PESP modification.

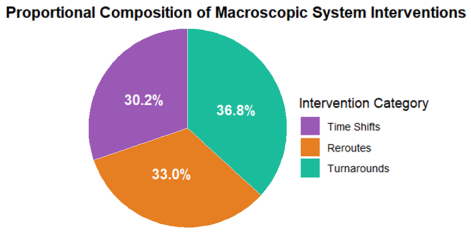


**Figure 6.5:** Proportional Composition of Macro Adjustments.

Breaking down these changes in PESP across the five test cases in Figure 6.6 shows that there are preferences between test cases. The test `case Dordrecht_lagezwaluwe` shows the lowest intensity, with approximately the same number of changes for each type. The `sector ede _arnhem` shows a clear inclination toward turnaround operations (*keringen*) and time shifts (*tijdsverleggingen*). The highest intensity can be found in the test `case in Rotterdam _gouda _2027`, which can be explained by the location of the maintenance, which goes directly through the Randstad.

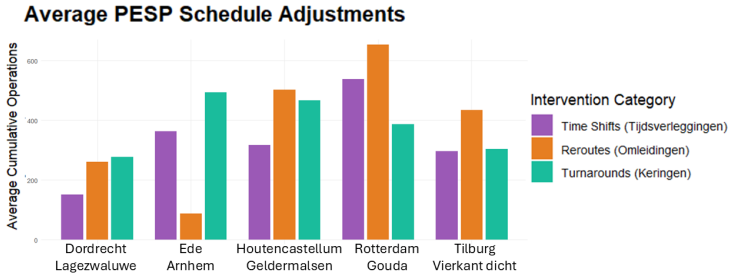


**Figure 6.6:** Macro Adjustments Aggregated by DRP Scale.

Finally, evaluating the interventions against station size classes in Figure 6.7 exposes significant difference in distribution. The smaller DRPs have almost no reroutes and minimal time shifts or turnarounds due to their rigid track topologies. Medium stations handle the highest absolute volume of network changes, distributed almost equally across all three types of modifications.

In contrast to the medium hubs, large hubs display different image: time shifts (*tijdsverleggingen*) skyrocket as the popular change type, while reroute and turnaround frequencies fall below the levels recorded at smaller medium stations. This shows that at large stations,

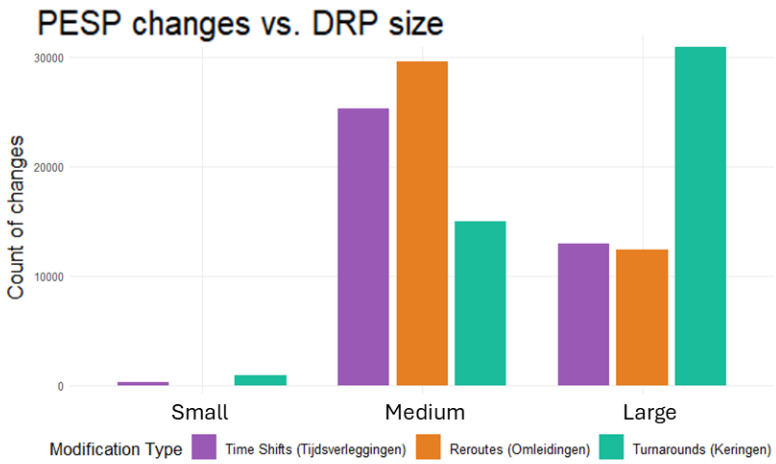


**Figure 6.7:** Mean Cumulative Adjustments per Active Run Window.

the scheduling system avoids changing the physical routes of the trains to prevent massive traffic jams. Instead, it simply delays trains to allow the crowded platforms to clear up.

After looking at the absolute numbers of conflicts, the PESP changes in the pie chart, and which modifications are preferred, the next step in the data exploration is to verify the distribution of the actual conflicts. Since actual conflicts occur in a much smaller group, understanding their distribution and how they relate to other parameters is important for optimising the number of micro computations. Figure 6.8 shows the four graphs used to check this normality and linearity.

Looking at the first plot in the upper-left corner, the zero inflation is clearly visible. A large part of the DRPs are compliant nodes to the PESP modifications, meaning they have zero conflicts. The red line shows a slight fluctuation below 10, indicating that the DRPs that actually have conflicts usually have a maximum of 10. In the upper-right figure, a Q-Q plot is displayed. This plot is used to determine whether a distribution follows a normal distribution. If it were normal, the red data points would closely follow the black line. What is actually visible is that the data deviates significantly above zero and shows almost an exponential peak. This is a clear sign that the data has no normality. Another distribution to check is the Poisson distribution. The reason for checking this is that Poisson handles discrete count values well and can capture the "tipping point" that a DRP experiences. The graph in the lower-left shows the actual data compared to a Poisson fit, and it is clear that the Poisson distribution follows the data very well. This gives a clear indication that the data are much more Poisson than normal. To confirm the Poisson distribution, an easy method is to calculate the mean and variance, which should be close

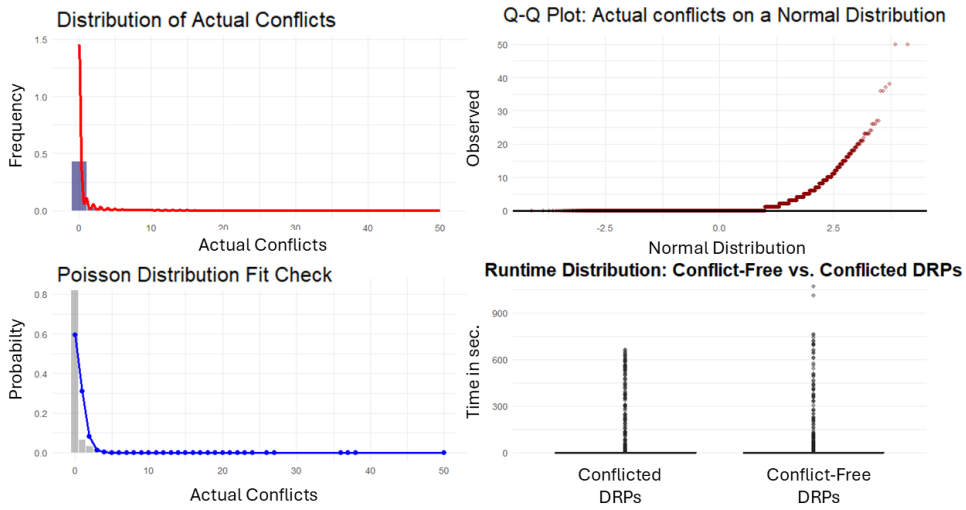


**Figure 6.8:** Exploration of the Normality and Linearity of the Actual Conflicts

to equal. Reviewing these data indicates a mean of 0.359 and a variance of 2.744, which are not similar. Considering the variance is much higher, the data is overdispersed, meaning we should look at models closely related to Poisson that can handle this, like Quasi-Poisson and Negative Binomial. The lower right graph displays the runtime distribution. In this graph, two boxplots are shown: the left one represents the distribution of runtimes when the DRP cannot be solved to a conflict-free level, while the right one indicates the DRPs that can be resolved to a conflict-free state. An interesting observation is that both distributions are quite similar, and the DRPs that can be solved do not consistently show significantly lower runtimes. This suggests that the SCM algorithm does not arrive at a conflict-free solution much faster than in cases where a feasible resolution is not possible. Overall, moving from absolute counts and PESP preferences to analyzing the distribution itself confirms that the conflict data is non-normal, heavily zero-inflated, and overdispersed. Uncovering these specific patterns and non-linear trends provides a clear picture of when and why conflicts occur, which is important for understanding how to optimise the communication between PESP and micro and ultimately decrease the running time.

# 7

# Algorithmic Optimization

The automated scheduling model works as an iterative alternating loop. It computes an alternative schedule and passes down the DRPs that have changed to the microlevel where the SCM algorithm is used to solve the DRP with the PESP changes. The model iterates between the two until both levels are feasible. Currently, communication between the two levels is highly static. The macro layer passes the modified DRP and if the SCM algorithm cannot solve a DRP to conflict free level, it gives back constraints to the macro level to address this. This architectural layout imposes a large number of computations each iteration and when the PESP schedule is highly infeasible on micro level, each micro computation is redundant. To achieve the performance target of this research, which is minimizing the global execution runtime, the main point of adjustment must be to target this communication layer.

## 7.1 Filtering Conflict-Free Nodes

The subset of changed DRPs that are received from the PESP output can still be quite a large group to evaluate. The range of modified DRPs frequently goes up to 300. Running the SCM algorithm for each of these DRP is computationally intensive, and given that not each DRP gives back constraints to PESP due to the unsolvability, this method runs a lot of unnecessary computations. To decrease the number of SCM calculations, a pre-filter is implemented to check if a location actually contains conflicts before running the full optimization model, this is visualized in figure 7.1 in the first step directly after the PESP model called **Filtering**.

To perform this check, for each changed DRP, an SCM environment is built by loading in all track possession times. In this stage, there are no platform assignments given and the

possible conflicts are decided by overlapping track occupancies (*bezettingen*) by trains. By mapping these occupancies, the number of potential conflicts can be determined, specifically for platform and crossover conflicts. For each of the possible conflicts, the ignorable check is performed. In this model the original schedule is assumed as conflict free even if there are conflicts. Therefore whenever one of the possible conflicts is the exact same as a conflict in the current schedule it can be ignored. By reducing the number of possible conflicts with ignorable conflict, the number of DRPs that undergo SCM decreases even further.

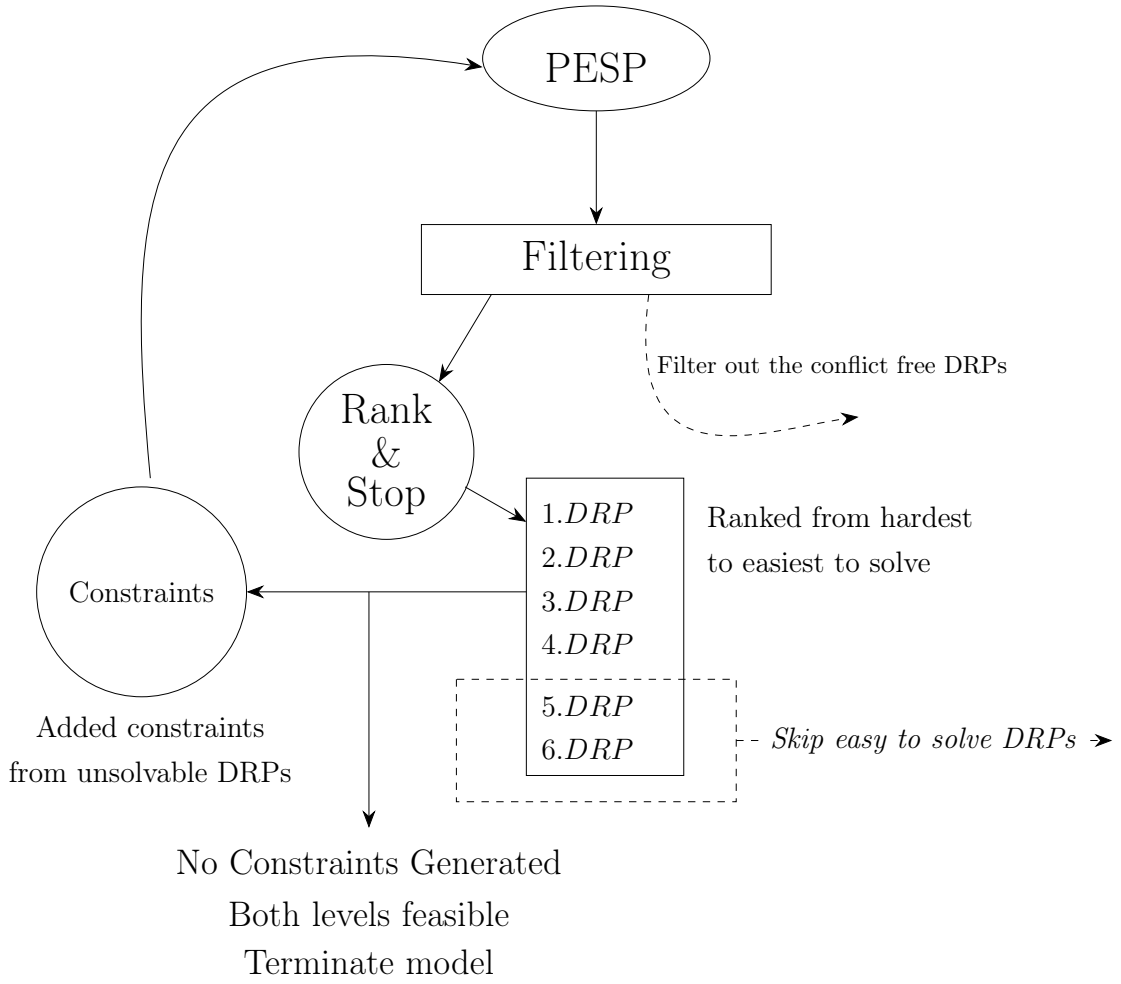


**Figure 7.1:** Algorithmic Optimization Workflow: Integrating Filter, Rank and Stop Method.

## 7.2 Empirical Risk Profile

Furthermore, to assist the prediction model, historical data about the DRPs is used to give a risk score. DRPs that have a high probability of being unsolvable regardless of the maintenance case should be pushed up in the list, as the certainty of solving is very low, in contrast to DRPs that remain conflict-free in most scenarios.

To account for this, an empirical risk profiling approach is used. Specifically, a target encoding technique is applied, which transforms each DRP (categorical station identifier) into a numerical value based on its observed failure rate. In this context, target encoding replaces each station with the historical probability that it leads to an unsolvable conflict, thereby capturing its predictive relationship with the outcome variable.

However, raw target encodings are highly unstable for low-frequency stations, where limited observations can lead to extreme and noisy estimates. To mitigate this, Bayesian (or M-estimate) smoothing is applied. This method blends the station-specific failure rate with the global network failure rate, effectively shrinking unreliable estimates toward the overall mean and reducing overfitting. This results in the following smoothed empirical risk score ($S_j$):

$$S_j = \frac{C_j + (m \cdot \mu)}{N_j + m} \tag{7.1}$$

Where $C_j$ is the count of verified unsolvable failures at station $j$, $N_j$ is the total number of occurrences of station $j$ within the training partition, $\mu$ is the global baseline failure rate of the network, and $m$ is the smoothing weight (configured to $m = 8$). The parameter $m$ acts as a pseudo-count that controls the strength of the global prior: small-sample stations are strongly pulled toward the network average, while high-frequency stations retain their empirical behavior.

## 7.3 Priority-Based Queue Ranking Optimization

Once conflict-free nodes are filtered out, the model is left with a subset of potentially conflicted DRPs. In the baseline approach, these locations are processed in a random sequence, where each SCM is executed to completion. If an SCM is infeasible, constraints are returned to the macro-layer (PESP model), potentially triggering a rollback and a new iteration. This approach often results in unnecessary computation, as many solvable DRPs are processed before encountering a critical infeasible one.

To address this inefficiency, this research introduces a priority-based ranking mechanism that orders DRPs according to their predicted difficulty. The key distinction lies between solvable and unsolvable DRPs. Solvable DRPs can be resolved within the micro-layer through platform reassignment without altering the macro schedule, while unsolvable DRPs represent structural bottlenecks that require macroscopic interventions, such as train cancellations or rerouting. These interventions can only be triggered through constraints returned to the PESP model.

By ranking DRPs from highest to lowest predicted difficulty, the model prioritizes the processing of critical, unsolvable instances. This is shown in figure 7.1 by the **Rank & Stop** circle. This allows the system to extract the necessary macro-level constraints early in the microscopic stage. Once all unsolvable DRPs have been identified, an early-stopping criterion is applied: the remaining solvable DRPs are skipped, as they do not contribute additional constraints. This avoids redundant SCM computations and directly reduces iteration runtime.

This prioritization strategy is particularly effective when high-risk DRPs are associated with infrastructure zones characterized by long occupation times and tight bottlenecks. Resolving these dominant constraints early significantly restricts downstream space-time possibilities, thereby accelerating convergence of the overall system (17). In contrast, simple and compliant DRPs are deferred until the final convergence stage, preventing them from consuming unnecessary computational resources during intermediate iterations.

In this research, seven different predictive metrics are outlined. Each model evaluates the operational data using different mathematical structures, offering different methods to relax or tightenj the priority scoring.

### 7.3.1 Linear Baseline

The first ranking metric serves as a baseline, using a standard Ordinary Least Squares (OLS) linear regression model. This approach fits a linear combination of the extracted features of a DRP to output a continuous difficulty score, eliminating the need for hard-coded weight assignments by dynamically extracting data-driven coefficients ($\beta_i$). In this formulation, $\hat{y}$ represents the predicted difficulty score for a given DRP, $\beta_0$ is the intercept parameter, $n$ is the total number of extracted features, and $X_{ki}$ represents the specific value of feature $k$ for DRP $i$. The model is specified as follows:

$$\hat{y} = \beta_0 + \sum_{k=1}^{n} \beta_k X_{ki} \tag{7.2}$$

The model assumes that none of the indicators has significant interactions and the significance tests determine the exact features to be used in later experiments.

### 7.3.2 Distribution-Specific Generalized Linear Models

Standard linear models assume that data errors follow a normal (Gaussian) distribution. However, the conflict data are skewed and discrete. To address this, this section introduces three specialized GLMs. Instead of forcing a normal distribution onto the data, these models assume other distributions. They match the mathematical assumptions of the model to the actual shape of the data, such as the count or the bounded distributions. They use non-linear link functions to enforce physical constraints, matching the model's underlying probability distribution to the actual shape and variance of the data.

#### Negative Binomial Mixed Model

To capture the discrete and heavily overdispersed nature of the network data, the second metric implements a Negative Binomial Mixed Model. Instead of assuming normal errors, this generalized linear mixed model (GLMM) adapts to a negative binomial distribution, which is explicitly designed for count data where the variance exceeds the mean (37). It accounts for interaction effects between continuous and categorical predictors while separating fixed population trends from cluster-specific random variation. This enables the model to capture both global trends and station-level deviations by incorporating a random intercept per group $j$. The model relates the expected response to the predictors via a logarithmic link function:

$$\log(\mu_{ij}) = \beta_0 + \sum_{k=1}^{P} \beta_k X_{kij} + \sum_{m=1}^{Q} \gamma_m Z_{mij} + \sum_{k=1}^{P} \sum_{m=1}^{Q} \delta_{km}(X_{kij} \times Z_{mij}) + u_j \tag{7.3}$$

where:

- $\mu_{ij} = E(y_{ij})$ is the expected count outcome for observation $i$ in group $j$.
- $\beta_0$ is the global fixed intercept.
- $X_{kij}$ and $Z_{mij}$ are continuous and categorical predictors, respectively.
- $\beta_k$, $\gamma_m$, and $\delta_{km}$ are the fixed-effect coefficients.
- $u_j \sim N(0, \sigma_u^2)$ is the random intercept for group $j$.

The conditional variance of the outcome follows a quadratic overdispersion structure, defined as $\text{Var}(y_{ij} \mid u_j) = \mu_{ij} + \frac{\mu_{ij}^2}{\theta}$, where $\theta$ is the dispersion parameter. This parametric form directly accommodates the uncharacteristically long tail of the data without requiring arbitrary transformations.

### Quasi-Poisson Mixed Model

This count-based multi-level approach is further expanded by the third model, the Quasi-Poisson Mixed Model, which focuses directly on correcting uncertainty windows and capturing operational tipping points. Similarly to the Negative Binomial framework, it utilises a log-link function to model discrete counts and enforces positive predictions. However, rather than assuming a strictly defined probability distribution, the Quasi-Poisson model uses a quasi-likelihood approach where the variance is modeled as a direct linear function of the mean multiplied by a constant dispersion factor (38):

$$\log(\mu_{ij}) = \beta_0 + \sum_{m=1}^{M} \beta_m X_{mij} + \sum_{n=1}^{N} \gamma_n Z_{nij} + \sum_{m=1}^{M} \sum_{n=1}^{N} \delta_{mn}(X_{mij} \times Z_{nij}) + u_j \tag{7.4}$$

where:

- $\mu_{ij} = E(y_{ij})$ is the expected count outcome for observation $i$ in group $j$.
- $X_{mij}$ and $Z_{nij}$ are the continuous and categorical predictors, respectively.
- $\beta_m$, $\gamma_n$, and $\delta_{mn}$ are the fixed-effect coefficients.
- $u_j \sim N(0, \sigma_u^2)$ is the random intercept for group $j$.

The key characteristic of this model is its variance structure, defined as $\text{Var}(y_{ij} \mid u_j) = \phi\mu_{ij}$, where $\phi$ represents the empirical dispersion parameter. When $\phi > 1$, the model dynamically scales up the standard errors of the fixed effects. This directly compensates for the high volatility found in the data, preventing overconfident baseline predictions and reducing false-positive significance flags.

### Quasibinomial Bounded GLM

The fourth architecture addresses zero-inflation and bounded limits using a Quasibinomial Bounded Generalized Linear Model (GLM). Here, the actual conflict target is mapped onto a fractional ratio that falls into the interval $[0, 1]$. To accommodate these strict boundaries, the model uses a logit link function to estimate an empirical dispersion parameter ($\Phi$).

This quasi-likelihood framework handles overdispersion and accommodates zero-inflation without enforcing a strict Poisson structure or predicting mathematically impossible values outside the bounded range:

$$\ln\left(\frac{\mu_i}{1-\mu_i}\right) = \beta_0 + \sum_{k=1}^{P}\beta_k X_{ki} + \sum_{m=1}^{Q}\gamma_m Z_{mi} + \sum_{k=1}^{P}\sum_{m=1}^{Q}\delta_{km}(X_{ki}\times Z_{mi}) \tag{7.5}$$

$$\mathrm{Var}(Y_i) = \Phi\mu_i(1-\mu_i) \tag{7.6}$$

where:

- $\mu_i = E(Y_i)$ is the expected fractional outcome for observation $i$.
- $X_{ki}$ and $Z_{mi}$ are the continuous and categorical predictors, respectively.
- $\beta_0, \beta_k, \gamma_m$, and $\delta_{km}$ are the estimated fixed-effect coefficients.
- $\Phi$ is the quasi-likelihood dispersion parameter capturing overdispersion.

Priority scoring can be tightened within this framework by decreasing the probability classification cut-off point ($\mu_i$), thereby forcing the model to be more aggressive in flagging minor stations as high-risk nodes.

### 90th Percentile Quantile Model

To focus specifically on catastrophic bottlenecks rather than average network behavior, the fifth architecture uses a 90th Percentile Quantile Model. The difference between this model and all previous models is that the previous models aim to predict the average outcome. While doing this, the models miss what happens during worst-case scenarios. The 90th Percentile Quantile Model concentrates only on the worst 10%. This is achieved by optimizing the parameters around specific conditional quantiles rather than the conditional mean, thereby modeling different regions within the response distribution instead of a single average relationship (39). It fits an optimization plane directly to the extreme operational tail ($\tau = 0.90$) where severe gridlock occurs. The model does not need assumptions about the data's shape or variance. Instead, it gives a clear overview of how the predictors behave when the system is under extreme stress compared to when it runs normally.

$$\min_{\beta}\sum_{i=1}^{n}\rho_\tau\left(y_i - \left[\beta_0 + \sum_{k=1}^{P}\beta_k X_{ki} + \sum_{m=1}^{Q}\gamma_m Z_{mi} + \sum_{k=1}^{P}\sum_{m=1}^{Q}\delta_{km}(X_{ki}\times Z_{mi})\right]\right) \tag{7.7}$$

$$\rho_\tau(r) = r\left(\tau - \mathbb{I}(r<0)\right) \tag{7.8}$$

where:

- $\beta$ represents the complete vector of parameters being optimized by the model.
- $\beta_0$ is the baseline intercept coefficient, and $\beta_k$ represents the regression coefficients assigned to each continuous feature.
- $\tau = 0.90$ represents the targeted 90th percentile conditional quantile.
- $y_i$ is the observed response variable for observation $i$.
- $X_{ki}$ and $Z_{mi}$ are the continuous and categorical predictors, with coefficients $\beta_k$, $\gamma_m$, and $\delta_{km}$.
- $\rho_\tau(r)$ is the loss check function penalizing residuals $r$ asymmetrically based on $\tau$.
- $\mathbb{I}(\cdot)$ is the indicator function evaluating to 1 if the condition is true, and 0 otherwise.

where $\rho_\tau(r) = r(\tau - \mathbb{I}(r < 0))$ represents the asymmetric absolute loss check function. Shifting the quantile parameter $\tau$ upward tightens the model focus to isolate only the most severe conflicted DRPs, while lowering $\tau$ relaxes it toward the median.

### 7.3.3 Non-Parametric Machine Learning Ensembles

Unlike the previous parametric architectures that rely on distributional assumptions and linear forms, machine learning models are non-parametric. They avoid predefined mathematical curves, using data-driven algorithms to discover structural patterns directly from the observed data. Including these models has several purposes. First, they detect hidden interactions and non-linear thresholds automatically and therefore eliminate the need to specify the interaction terms. Second, they give the upper bound in prediction performance; if they significantly outperform the GLMs, it proves that conflict prediction is led by complex nonlinear dynamics. Lastly, because they cannot predict values that fall outside of the training data they manage overdispersion and bounds naturally.

#### Random Forest Model

The sixth architecture implements a Random Forest model, which achieves this non-parametric flexibility by constructing a parallel collection of $N = 300$ de-correlated decision trees using bootstrap aggregation (40). In this ensemble formulation, $\hat{y}$ represents the final predicted difficulty score, $N$ is the total number of decision trees in the forest, $x$

denotes the input vector of extracted station features, and $T_b(x)$ represents the individual prediction output by the $b$-th decision tree. The aggregated model is specified as follows:

$$\hat{y} = \frac{1}{N} \sum_{b=1}^{N} T_b(x) \tag{7.9}$$

By averaging predictions across an ensemble where each split is restricted to a random subset of features, the model limits overfitting. This parallel, randomized structure allows it to organically trace flat, zero-inflated baselines and jagged capacity boundaries that traditional curves miss.

**Extreme Gradient Boosting**

Alternatively, the seventh model utilizes an Extreme Gradient Boosting architecture (XGBoost). Unlike the independent and parallel approach of the Random Forest, XGBoost builds decision trees sequentially. It optimizes performance by fitting each new tree ($f_t$) directly to the negative gradients (errors) of the preceding models. In this additive formulation, $\mathcal{L}^{(t)}$ represents the objective function minimized at iteration $t$, $n$ is the total number of observations, $l$ is a differentiable loss function measuring the distance between the actual target value $y_i$ and the prediction from the previous step $\hat{y}_i^{(t-1)}$, $f_t(x_i)$ is the prediction of the newly added tree given the feature vector $x_i$, and $\Omega(f_t)$ is a regularization term that penalizes tree complexity. The objective function at step $t$ is specified as follows:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{7.10}$$

By adding trees step-by-step to correct previous mistakes and incorporating a regularized objective function ($\Omega$) to penalize model complexity (41), XGBoost typically achieves higher predictive accuracy and sharper boundary detection than parallel ensembles.

## 7.4 The Early Stopping Approach

Running a sorted DRP list to completion yields the exact same computational time as an unsorted run. The optimization comes from the intelligent early-stopping mechanism. By detecting the turning point of unsolvable DRPs to solvable DRPs, easy and compliant SCM computations can be skipped during the intermediate iterations. This method gives the model the ability to focus on the problem areas and reduce its total time. To ensure that the unsolvable DRPs are not skipped, a *certainty coefficient* is implemented that

determines when the deadlock is triggered while maintaining a low probability that the skipped DRPs were unsolvable.

In this research, both static and dynamic thresholds for the certainty boundary are evaluated. Static thresholds utilize fixed absolute or percentile cutoff marks, establishing rigid baseline rules. To address their inherent operational limitations under fluctuating conflict volumes, dynamic metrics are introduced to adapt directly to the sequential data stream. In total, six mathematical frameworks are evaluated to establish the early-stopping boundary.

### 7.4.1 Static Stopping Baselines

#### The Consecutive Solves Streak Metric

The consecutive solves streak metric represents the most straightforward method. It stops and deadlocks the moment a fixed predetermined number of consecutive solvable nodes ($x$) are successfully confirmed in the queue, immediately stopping the execution and returning the discovered constraints to the macro solver. While this metric is computationally simple, it relies heavily on the stationarity of the queue lengths. However, because the size of the active conflict set fluctuates between 20 and 300 DRPs depending on the maintenance layout, a static absolute metric could fail; too low a threshold reduces the large sets too much, while too high a threshold will allow the entire list of DRPs to be computed, completely eroding the computational savings.

#### The Sliding Percentile Metric

Instead of looking at the absolute value, the percentile metric uses a relative value. Instead of counting raw nodes, this metric calculates the running proportion of all processed stations that the micro model has found to be unsolvable up to index $t$, which is defined as $\rho_t$ and computed as follows:

$$\rho_t = \frac{1}{T} \sum_{i=1}^{t} \mathbb{I}(y_i = \text{Unsolvable}) \tag{7.11}$$

where $\mathbb{I}$ is an indicator function and $y_i$ is the solvability status of the ith DRP in the queue. The stopping rule triggers a deadlock when $\rho_t$ drops below a fixed percentile floor ($\rho_{\min}$). Although less impacted by different queue sizes, it does still assume linearity and assumes that a larger set will have a larger number of unsolvable DRPs. In addition, the historical weight of the early unsolvable nodes can artificially inflate $\rho_t$, concealing a sudden localized transition into entirely solvable territory.

### 7.4.2 Dynamic Decision Metrics

#### The Moving Batch Window Metric

The moving batch window metric divides the total queue into smaller sets of DRPs and thereby eliminates the bias of full-queue percentiles. The DRP sequence ranked is evaluated through a fixed size sliding window $W$. The local failure density ($\rho_{\text{batch}}$) is computed exclusively within the active window slice at the current queue position $t$, where $y_i$ represents the solvability state of the DRP evaluated at step $i$ in the processing queue. The metric is formulated as follows:

$$\rho_{\text{batch}}(t) = \frac{1}{W} \sum_{i=t-W+1}^{t} \mathbb{I}(y_i = \text{Unsolvable}) \tag{7.12}$$

An early stop is executed when $\rho_{\text{batch}}(t)$ drops below a dynamically scaled threshold. This offers the volatile, top-ranked DRPs maximum operational leeway to expose their constraints early in the sequence, while systematically tightening the stopping bounds as the local lookback window encounters long, uninterrupted blocks of compliant, solvable stations.

#### The Sequential Probability Ratio Test

The Sequential Probability Ratio Test (SPRT), originally formulated by Abraham Wald (42), frames early stopping as a continuous statistical hypothesis test between two competing stochastic processes. Unlike fixed-sample tests, SPRT evaluates data points sequentially, producing three possible outcomes at each step $t$: the null hypothesis is rejected, it is accepted (terminating the process), or the evidence remains insufficient to draw a conclusion, leading to the evaluation of the next queue item.

In this framework, the null hypothesis $H_0$ states that the queue is operating within the *Constraint-Harvesting Zone*, where the probability of encountering an uninformative solvable node is low ($H_0 : p = p_0$). In contrast, the alternative hypothesis $H_1$ states that the queue has crossed into the *Information Depletion Zone*, where the probability of encountering a solvable node is high ($H_1 : p = p_1$).

As each node $t$ is unmasked, the cumulative log-likelihood ratio $\Lambda_t$ is updated recursively based on the observed binary outcome $y_t \in \{0, 1\}$, where $y_t = 1$ denotes a clean solution and $y_t = 0$ denotes an informative meltdown:

$$\Lambda_t = \Lambda_{t-1} + \ln\left(\frac{P(y_t \mid H_1)}{P(y_t \mid H_0)}\right) \tag{7.13}$$

The step-wise updates adjust $\Lambda_t$ symmetrically according to the direction of classification. When a clean solve ($y_t = 1$) is unmasked, a positive step-size of $\ln(p_1/p_0)$ is added to the score, representing accumulated evidence in favor of $H_1$. In contrast, when a constraint-generating meltdown ($y_t = 0$) is unmasked, a negative step-size of $\ln((1-p_1)/(1-p_0))$ is applied, penalizing the score and pulling the walk away from the stopping boundary.

Wald's framework establishes strict mathematical boundaries governed by two predefined error tolerances, which function as operational risk parameters in this active queue setting:

- **Type I Error ($\alpha$):** Represents the probability of a False Alarm. In this context, $\alpha$ is the probability of prematurely concluding that the queue has entered the uninformative zone ($H_1$) when it is actually still traversing a dense pocket of meltdowns ($H_0$). Tightening $\alpha$ raises the stopping wall to protect against information leakage and missed constraints.

- **Type II Error ($\beta$):** Represents the probability of a Missed Signal. Here, $\beta$ is the risk of failing to recognize that the valuable constraint harvest has ended, mistakenly assuming the process is still in the meltdown zone ($H_0$). Tightening $\beta$ lowers the threshold, forcing the engine to execute more solves to maintain strict confidence, whereas increasing $\beta$ decreases caution and prioritizes computational savings.

The model terminates queue execution and triggers an operational cutoff at the exact moment the cumulative ratio accumulates sufficient evidence to breach the upper stopping wall:

$$\Lambda_t \geq \ln\left(\frac{1-\beta}{\alpha}\right) \tag{7.14}$$

By tuning these error thresholds, the system explicitly controls the equilibrium between computational expenditure ($\beta$) and constraint leakage prevention ($\alpha$).

### Multi-Armed Bandit Upper Confidence Bound

The Multi-Armed Bandit problem is used in sequential decision-making when the distribution is uncertain. A popular example of MAB is the $k$ slot machines and an octopus that pulls on each slot machine and tries to maximize its cumulative reward. The octopus is unaware of the reward distributions of each slot machine and by balancing exploration and exploitation, it converges to the highest rewarding machines. The formal definition addresses the trade-off between *exploration* (discovery of new observations in unknown search regions) and *exploitation* (maximizing highly rewarding areas) (43, 44). As noted

by Katehakis and Veinott (45), optimizing this space requires a mechanism that dynamically balances empirical rewards with allocation variance.

To redesign this method as an early-stopping task, the problem is framed as a finite-horizon bandit problem where the verified solvability of a node acts as the reward signal. This general principle of optimism is implemented directly via our deterministic, index-based allocation rule:

$$A_t = \arg\max_a \left[ Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right] \tag{7.15}$$

where $Q_t(a)$ represents the observed mean reward for action $a$ (continuing evaluation versus execution of a stop), $N_t(a)$ tracks the total frequency of that action and $c$ controls level of exploration. As the queue progresses and the allocation count $N_t(a)$ increases, this optimistic term systematically decays. This mechanism forces the confidence interval to shrink tightly around the true mean, naturally fading out exploration to execute a secure, risk-bounded operational stop the moment the unsolvable DRPs are all checked.

### Thompson Posterior 95% Limit

Thompson Sampling addresses the MAB dilemma by maintaining a continuous dynamic probability distribution over the parameters of the reward system, rather than relying on a static cutoff index like UCB (46). As explained by Russo et al. (47), the benefit of Thompson Sampling lies in its ability to *"allocate observations according to their posterior probability of being optimal,"* naturally blending prior domain knowledge with real-time evidence.

For our early-stopping queue, this framework functions by creating a dynamic "belief curve" that follows a joined Beta-Binomial distribution. The parameters of this curve are determined at each step by the results of the past steps and an evolving margin of error. The system initializes a prior belief space anchored by two parameters, $\alpha_t$ and $\beta_t$:

$$\theta \sim \text{Beta}(\alpha_t, \beta_t) \tag{7.16}$$

where $\alpha$ represents the "wins" (verified solvable stations) and $\beta$ represents the "losses" (unsolvable bottlenecks). These parameters can be manually adjusted at the start to skew the model's baseline behavior. For example, initializing the system with a priori of $\beta_0 = 5$ means that the engine pretends that it is already five steps into the queue with five consecutive losses. This gives the model an operational headstart in failures, making it significantly more cautious and hesitant to stop early during the initial phases.

Once queue processing begins, the model dynamically updates its parameters after every single station based on the verified operational result.

$$(\alpha_{t+1}, \beta_{t+1}) = \begin{cases} (\alpha_t + 1, \beta_t), & \text{if } y_t = 0 \text{ (Solvable)} \\ (\alpha_t, \beta_t + 1), & \text{if } y_t = 1 \text{ (Unsolvable)} \end{cases} \tag{7.17}$$

As these underlying parameters change, the shape and location of the belief distribution obviously shift.

Instead of executing a greedy stop based on the expected raw average, the final cutoff point is decided by a strict 95th percentile rule. By evaluating the inverse Beta cumulative distribution function ($q_\beta$):

$$\Omega_t = q_\beta(0.95; \alpha_t, \beta_t) \tag{7.18}$$

the framework monitors the upper boundary of the system's true failure risk. When the belief curve shifts and becomes so narrow that its 95th percentile drops below our tolerance floor, the model establishes a 95% statistical confidence boundary. At this precise point of high certainty, the engine can safely conclude that the remaining tail of the queue contains only solvable stations, triggering a secure and optimal early stop.

# 8

# Experimental Setup

In this chapter, each step is explained to determine which metric to choose in the optimization step. Provides a complete experimental design that evaluates the performance of predictive ranking models and adaptive early-stopping metrics under out-of-sample conditions.

## 8.1 Experimental Architecture and Data Partitions

The experimental pipeline is structured as an evaluation script written in R. To prevent any data leakage and ensure that the predictive mechanisms can generalize to completely unseen schedules, the total dataset ($N = 91,278$) is split using a *Group-Based Train/Test Partition Strategy*.

Instead of a randomized split, which would break the dependency between DRPs due to geographical proximity, an entire maintenance testcase footprint is isolated and held out to form the testing partition. Throughout the experiments each step is tested on a new testcase and trained on the previous test sets.

Continuous characteristics are normalized using non-parametric boundary transformations. Empirical Cumulative Distribution Functions (ECDF) are generated from the training slice and mapped onto both partitions, bounding variables like `BUSYNESS`, `keringen`, and `possible_conflicts` within a standard $[0, 1]$ percentile limit.

## 8.2 Feature Significance Verification Tests

The feature selection is performed for all difficulty scoring method both parametric and non-parametric model families to maintain a even design:

- **OLS Baseline:** Uses a stepwise elimination framework to drop insignificant features, measuring changes in the Akaike Information Criterion (AIC) to optimize the model.
- **Negative Binomial, Quasi-Poisson and Quasibinomial GLM Feature Analysis:** Evaluates feature significance using an **Analysis of Deviance $F$-Test**. This mathematically accounts for the estimated dispersion parameter ($\Phi$) used to manage the data's zero-inflation.
- **90th Percentile Quantile Regression Evaluation:** Uses a **Joint Wald Test** with bootstrapped standard errors to isolate which predictors are significant at the extreme operational tail ($\tau = 0.90$).
- **Machine Learning Feature Permutation Screen:** Verifies variable importance for the Random Forest ($N = 500$ trees) and Gradient Boosting (XGBoost) models by checking their Mean Decrease in Impurity (MDI) profiles.

## 8.3 Model Performance Evaluation Measures

Predictive metrics are scored against the isolated test case. Their performance is tracked with five core measures:

1. **Out-of-Sample Spearman Rank Correlation (Spearman's $\rho$):** Measures how accurately the model sorts the queue from hardest to easiest. It checks the alignment between predicted difficulty scores and true conflict counts.
2. **Classification Area Under the ROC Curve (AUC):** Measures the model's ability to distinguish between solvable stations ($y = 0$) and inherently unsolvable bottlenecks ($y > 0$) across all decision thresholds.
3. **Optimal Threshold Accuracy (Youden's $J$):** Determines the best operational cutoff point on the ROC curve by maximizing the balance between sensitivity and specificity:
$$J = \text{Sensitivity} + \text{Specificity} - 1 \tag{8.1}$$
4. **False Negative Rate (FNR):** Tracks the most critical error state—when an unsolvable station is incorrectly flagged as solvable. This measures the risk of letting missed constraints leak past the early filters.

Standard classification metrics such as accuracy, AUC, and false negative rate (FNR) evaluate errors equally, which means that all errors are treated the same. The penalty of a miss is dictated by its **queue displacement distance**. If the difficulty metric misclassifies an unsolvable DRP but sorts it near the top-priority boundary, the early-stopping mechanism encounters it early and adapts. Conversely, if a high-conflict bottleneck is buried deep within the low-priority queue under the assumption that it is safe, the mechanism wastes significant computational time computing compliant DRPs before the unsolvable is reached.

To measure this sequence-dependent risk, this thesis introduces a custom **Positional Penalty Score ($PPS$)**. Let $N = 100$ represent the number of DRPs within a simulation run. The priority threshold, $K$, is dynamically assigned as the true count of the conflicted instances. In perfect ranking, the conflicted DRPs all rank between 1 and K.

To validate if the conflicted instance is within the K rank, let $R_{\text{actual}}(i)$ denote its true conflict-based rank, and $R_{\text{pred}}(i)$ denote the rank assigned by the predicted difficulty score of the model. The positional penalty is formulated as:

$$PPS = \sum_{i \in U} \max\left(0, R_{\text{pred}}(i) - R_{\text{actual}}(i)\right) \times \text{Actual Conflicts}_i \tag{8.2}$$

The $\max(0, \cdot)$ operator acts as an **asymmetric gate**:

- **Safe Overestimation:** If a model overestimates risk and ranks a bottleneck higher than its true position ($R_{\text{pred}} \leq R_{\text{actual}}$), the penalty is 0. The model is not penalised for being risk-averse.
- **Fatal Underestimation:** If a model underestimates risk and buries a conflicted DRP past the dynamic cutoff line ($R_{\text{pred}} > K$), the penalty scales linearly with the displacement distance and is multiplied by the number of actual conflicts.

This metric provides a zero-sum validation window. If a model wastes a high-priority slot on a zero-conflict station, it forces a true conflicted station to fall below the $K$ safety line. The model with the lowest accumulation $PPS$ represents the safest operational choice to protect the PESP loop from execution delays.

The outcome of the performance tests is shown in a table and is used to make the decision on the difficulty ranking metric.

## 8.4 Hyperparameter Tuning and Multi-Case Grid Search

To further enhance the performance of the certainty metrics, grid search is used to find the best parameter combinations for each. The search space evaluates each parameter combination in the given ranges, and assesses each individual scenario queues to isolate optimal thresholds. The objective of the grid search is a custom **Fitness Objective Function** . This function includes both objectives: reducing time and gathering as many constraints as possible.

$$\text{Fitness} = (\text{SAVED\_REWARD} \times \text{Avg\_Saved}) - (\lambda_{\text{PENALTY}} \times \text{Avg\_Missed}) \tag{8.3}$$

It applies a penalty parameter ($\lambda = 3$) to any combination of parameters that allows a macro-level constraint to slip through, and it gives a reward (SAVED_REWARD = 2)to any solvable DRP that is skipped. It has a slightly stronger penalty weight to account for the imbalance in the solvable-to-unsolvable ratio.

The exact hyperparameter combinations evaluated across the six independent grids and the specific ranges are given in table 8.1.

**Table 8.1:** Grid Search Parameter Dimensions per Certainty Metric

| **Metric** | **Hyperparameter Grid Range** | **Optimization Goal** |
|---|---|---|
| Consecutive Solves | `max`: $c \in \{5, 7, 9, 11, 13, 15\}$; `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Isolate strict baseline solve streaks. |
| Sliding Percentile | `rate`: $\rho \in [0.05, 0.35]$ (step 0.05); `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Identify long-run failure density floors. |
| Moving Window | `size`: $W \in \{5, 10, 15, 20, 30\}$; `rate`: $\rho \in [0.0, 0.2]$ (step 0.05); `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Track local information depletion spikes. |
| Statistical SPRT | `p0`: $[0.10, 0.35]$; `p1`: $[0.65, 0.85]$; `err`: $\{0.01, 0.05, 0.10\}$; `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Map continuous log-likelihood ratio walls. |
| MAB UCB Explorer | `c`: $\{0.1, 0.2, 0.5, 1.0, 1.3, 1.5\}$; `bound`: $[0.30, 0.70]$ (step 0.05); `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Calibrate optimistic reward intervals. |
| Thompson Posterior | `prior`: $\alpha \in \{1, 2, 5\}$, $\beta \in \{1, 2, 5, 10\}$; `ceil`: $[0.70, 0.95]$ (step 0.05); `floor`: $f \in \{5, 10, 15, 20, 25\}$ | Tune dynamic posterior belief intervals. |

The optimal tuned parameters are extracted from this grid search and used for the final results.

## 8.5 The Robustness Sensitivity Analysis

To test the out-of-sample stability of the optimized configurations, a robustness sensitivity analysis is conducted across independent validation runs using the optimal difficulty metric. The purpose of this sensitivity analysis is to measure the degree to which the early-stopping metrics are consistently adjusted to different timetable conditions. Assess performance by tracking three key components: the number of missed constraints (data leakage), the percentage of skipped computations, and the number of wasted solves.

# 9

# Results

## 9.1 Data Distribution and Standardization

The initial exploratory data analysis revealed significant variance across the numerical predictors. As shown in the data chapter, a station's structural size directly shifts its distribution of PESP schedule modifications, while different test cases exhibit highly variable averages of possible conflicts. The high degree of zero-inflation within the distribution of actual conflicts leads to the inability of using the parameters in their original form. To address this imbalance, a standardization pipeline was implemented using the Empirical Cumulative Distribution Function (ECDF). A major benefit of using the ECDF is that it is a non-parametric transformation. Because the distribution of actual conflicts does not follow a normal distribution, it is important that the standardization does not assume normality either. Ultimately, the ECDF helps stabilize the variance of the continuous covariates before they are used for any model estimation.

## 9.2 Interaction Effects

To systematically evaluate the joint effects of network features on actual conflicts, several interaction tests were conducted. Testing for interaction effects is essential to identify whether the impact of a given independent variable is conditional upon the level of another. These dependencies were examined on three distinct dimensions: factor-by-factor, covariate-by-covariate, and factor-by-covariate configurations. Statistical testing revealed highly significant factor-by-covariate interactions between the categorical network size factor and three covariates: possible conflicts, ignorable conflicts, and total changes of PESP. To visualize this, the interactions are plotted in Figure 9.1. The significance in interaction

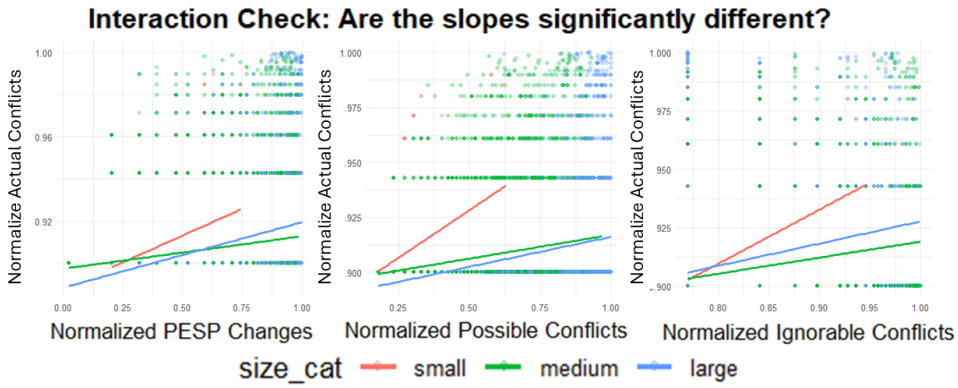


**Figure 9.1:** Interactions between Factor Size and the Covariates

can be seen by the difference in the slopes per size factor. This shows that the impact of the parameters have different levels of impact on the size factor.

All significant interactions are positive. This means that as a covariate (such as possible conflicts) increases, the number of actual conflicts also increases. However, the strength of this effect changes depending on the size of the network. Smaller station layouts are much more sensitive to these increases than medium layouts. This difference suggests that smaller, tighter rail layouts have less room for error, making their scheduling profiles much more volatile when changes occur. The smaller impact of the PESP modifications on the medium and larger DRPs can be explained by the fact that larger station layouts can absorb these changes more easily, offering more flexibility to move trains around. Having more platforms means the model has a wider range of alternative routing options to utilize.

## 9.3 Ranking Methods & Model Comparison

The first step in the "rank and stop" framework is to select the underlying difficulty metric. This research evaluates seven distinct models to predict and rank the DRPs: a linear baseline, a Negative Binomial mixed model, a Quasi-Poisson mixed model, a Quasibinomial bounded GLM, a 90th percentile quantile model, a Random Forest model, and an XGBoost model. Feature selection was conducted for each model independently, using testing methods tailored to their specific statistical structures. The final model specifications are detailed in Table 11.1 in the appendix.

The linear baseline is implemented as a full additive model with no interaction terms. Akaike Information Criterion (AIC) selection confirmed that the full model structure yielded the optimal fit for this linear baseline. Across all generalized linear models (GLMs)

and the 90th percentile quantile model, the interaction between the ignorable conflicts covariate and the size factor was dropped during feature selection due to the lack of statistical significance. In contrast, the empirical risk profile emerged as a very significant covariate in all evaluated models, confirming that including the historical risk in the framework was a good choice of methodology. To ensure a fair validation process, difficulty scoring models were trained in an isolated subset of maintenance scenarios and evaluated in a completely unseen test case. The comparative results of this experiment are summarized in Table 9.1.

| **Metric** | **Spearman $\rho$** | **AUC** | **Opt. Thresh.** | **Acc.** | **FNR** | **Dynamic PPS** |
|---|---|---|---|---|---|---|
| 90th QRM | 0.3878 | 0.8052 | 2.7077 | 0.7846 | 0.2550 | 1360.0000 |
| Linear Base | 0.4052 | 0.8168 | 0.8783 | 0.7801 | **0.1788** | 1131.4286 |
| XGBoost | 0.3642 | 0.7853 | 0.6588 | **0.8202** | 0.2808 | 1309.8571 |
| NB-GLMM | 0.4000 | 0.8115 | 0.2850 | 0.7995 | 0.2627 | 1047.5714 |
| QPoisson-M | 0.4015 | 0.8132 | 0.4983 | 0.7819 | 0.2378 | 1129.2857 |
| QBinomial | 0.4207 | 0.8282 | 0.4068 | 0.7899 | 0.2119 | 941.7143 |
| RF Model | **0.4509** | **0.8560** | 0.5371 | 0.7975 | 0.1887 | **896.8571** |

**Table 9.1:** Scoreboard for Difficulty Metrics. *Note: 90th QRM = 90th Percentile Quantile Regression Model; NB-GLMM = Negative Binomial Generalized Linear Mixed Model; QPoisson-M = Quasi-Poisson Mixed Model; RF = Random Forest. Best performing values per column are emphasized in bold text.*

As summarized in Table 9.1, each model is evaluated against the linear baseline model. Since the underlying scheduling dataset is non-linear, the linear baseline was expected to show the weakest performance. In all models, the impact of the empirical risk profile is clearly visible through an increase in the optimal classification thresholds, validating the importance of integrating historical DRP behavior. The Random Forest model achieved the highest Spearman rank correlation coefficient (0.4509), closely followed by the Quasibinomial GLM. This trend is mirrored in the AUC, where the Random Forest and Quasibinomial models again produce dominant results. In contrast, the accuracy metric presents an unexpected deviation: the highest overall score is achieved by XGBoost, a model that does not lead in any other evaluation rank. This can be explained by the fact that standard accuracy does not account for severe zero-inflation. In a highly imbalanced data set, a simple classifier that predicts "no conflict" can achieve a baseline accuracy of 80% or greater simply by matching the majority class. Hence, the False Negative Ratio and the Positional Penalty Score (PPS) carry significantly more weight. Surprisingly, the lowest False Negative Rate (FNR) is achieved by the linear baseline. This indicates a highly risk-averse behavior profile. However, the corresponding Spearman and PPS values reveal

its vulnerability: when the linear model misranks a DRP, the magnitude of the error is severe, which directly inflates the PPS value. This behavior is further emphasized by its optimal threshold, the second highest in the group, which demonstrates that it tends to aggressively misclassify solved DRPs as unsolvable, pushing them unnecessarily forward in the execution queue. The Random Forest model is the second-highest in the FNR scoring. Finally, the PPS column provides definitive evidence that the Random Forest and Quasibinomial models are the top two operational contenders by being the only two metrics dipping below the thousand points. Based on these insights, both models are selected for the second experiment, where they can be tested in combination with the certainty metrics.

## 9.4 Early Stopping Methods & Grid Search Optimization

To evaluate the early-stopping framework, six unique certainty metrics are analyzed. Each certainty engine undergoes an independent grid search optimization to find its peak hyperparameter set. The results are shown in Table 9.2 where the comparison between Random Forest and Quasibinomial is shown. The optimal parameter in the case of the Quasibinomial GLM reflects a conservative adaptation to a noisier ranking distribution. This is proved by a large optimized batch window size (30), a strict threshold of consecutive solves (9). In addition, the Thompson Posterior has a higher $\beta$ than $\alpha$ which means it models a start with more unsolved compared to solved stations to guard against an early termination due to the variance. Contrarily, the Random Forest parameters take advantage of a highly accurate ranking structure. Because Random Forest correctly sorts critical nodes and pushes them to the absolute front of the queue, grid optimization enforces a deep evaluation floor limit (25 steps) to clear the initial danger zone. Once this floor is passed, the system is allowed to use highly reactive metrics such as a small 5-node batch window and a low 5-consecutive-solve streak—to maximize computational savings without increasing risk.

To evaluate the performance of the stopping mechanisms, a simulation queue is tracked sequentially across the horizontal **Queue Steps** (x-axis), representing each DRP node sorted by predicted difficulty given by the ranking method. The plot utilizes a dual y-axis framework: the left axis measures the **Calculated Certainty Metric Value** (e.g., sliding percentile, batch window, MAB UCB, and Thompson limits), while the right axis represents **Raw Scale Metrics** multiplied by a factor of 10 to display unbounded integer counters (consecutive solve streaks and SPRT scores).

| Subsystem Stream | Hyperparameter Variable | Quasibin. GLM | Random Forest |
|---|---|---|---|
| **SPRT Engine** | Baseline Solvability Ratio ($p_0$) | 0.35 | 0.10 |
| | Target Solvability Ratio ($p_1$) | 0.85 | 0.65 |
| | Type I Error Boundary ($\alpha$) | 0.01 | 0.10 |
| | Type II Error Boundary ($\beta$) | 0.10 | 0.01 |
| | SPRT Evaluation Floor Limit | 5 | 25 |
| **MAB UCB Explorer** | UCB Exploration Rate ($c$) | 0.20 | 0.20 |
| | MAB Solvability Ceiling | 0.40 | 0.35 |
| | MAB Evaluation Floor Limit | 25 | 25 |
| **Thompson Posterior 95% Limit** | Beta-Prior Alpha Count ($\alpha_{prior}$) | 2 | 5 |
| | Beta-Prior Beta Count ($\beta_{prior}$) | 5 | 1 |
| | Thompson Solvability Ceiling | 0.75 | 0.70 |
| | Thompson Evaluation Floor Limit | 5 | 25 |
| **Consecutive Solves Streak** | Max Consecutive Solves Allowed | 9 | 5 |
| | Consecutive Evaluation Floor Limit | 25 | 25 |
| **Sliding Percentile** | Minimum Percentile Failure Density | 0.30 | 0.35 |
| | Percentile Evaluation Floor Limit | 25 | 25 |
| **Moving Batch Window** | Batch Window Evaluation Size | 30 | 5 |
| | Minimum Batch Failure Rate Limit | 0.20 | 0.20 |
| | Batch Evaluation Floor Limit | 5 | 25 |

**Table 9.2:** Side-by-Side Comparative Hyperparameter Configurations from Grid Search Optimization.

The underlying observed data is plotted as tracking nodes, where a green circle (`SOLVED`) denotes a conflict-free DRP and a red circle (`MELTDOWN`) an unfeasible, conflicted DRP. The shape and direction of each metric's line reflect how it reacts to these historical states over time. As the queue transitions from conflicted nodes into long streaks of compliant nodes, metrics that stop based on confidence in solvability move upward with each step, such as the Moving Batch Window, the MAB UCB Explorer, and the Thompson Posterior limit. Conversely, the remaining three metrics (the Sliding Percentile, Consecutive Solves streak, and Statistical SPRT Score) track the accumulation of failures, causing their lines to slope downward as more solvable stations are encountered. Finally, when the threshold for a stopping method is reached, a color-coded vertical line marks its termination, showing exactly where that method would stop the simulation. These resulting performance curves and termination points are visualized across both ranking scenarios in Figure 9.2 and Figure 9.3.

There are clear differences in how each model ranks the DRPs and how the resulting certainty metrics look. The Random Forest model in Figure 9.2 sorts data exceptionally well. It starts with a long sequence of unsolved DRPs and then slowly starts alternating

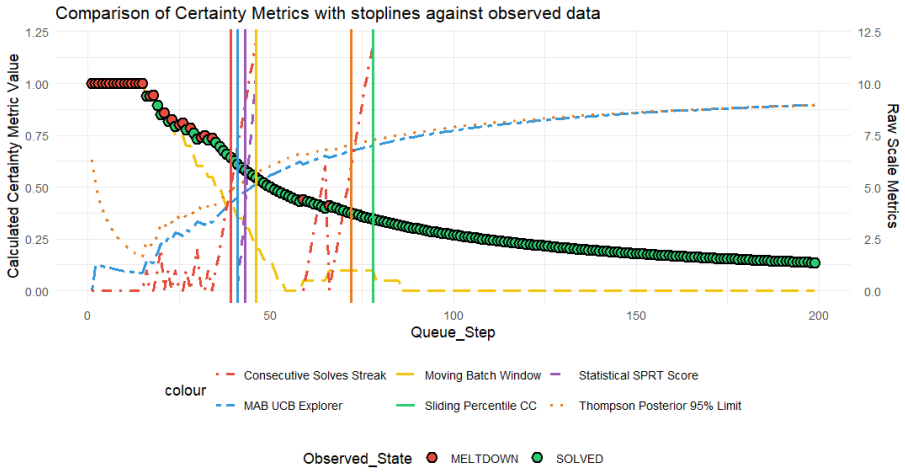


**Figure 9.2:** Random Forest Optimization Loop

between solved and unsolved steps. Furthermore, it has one long streak of solved DRPs before only two highly conflicted nodes end up misranked after many of the steps are solved.

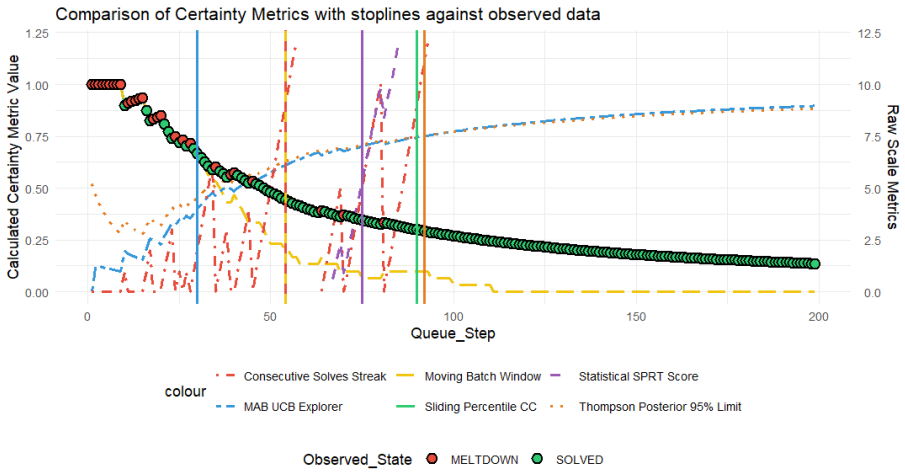


**Figure 9.3:** Quasibinomial GLM Optimization Loop

The Quasibinomial GLM model shown in Figure 9.3 also shows a good sorting pattern. It has a long block of meltdowns that is only slightly shorter than the one from the Random Forest. In its main risk area, it has small clusters of three to four unsolvable cases mixed with a few successes. This model manages to gather all failures before the 100th position in the queue, allowing the stop mechanism to stop before evaluating an additional 100 problems. This is a major decrease in the amount of computing power needed. However,

compared to the Random Forest model, which stops processing shortly after the 75th problem, the Quasibinomial shows slightly less efficiency compared to the Random Forest.

In this specific run, there are 27 unsolved cases out of 199 total DRPs. The order of the stopping lines differs between the two models. In the Random Forest (RF) case, the lines form two distinct groups, where four metrics stop near the beginning and two stop after the last unsolved DRP. In the Quasibinomial (QB) case, the stopping metrics are more spread out. This occurs because the QB model has a higher PPS value, which places more solvable DRPs near the top of the ranking and forces the metrics to be more cautious.

The shape of each metric reflects its underlying logic throughout the sequence. The Consecutive Solves Streak peaks whenever a solved DRP occurs, and when multiple solves happen in a row, the peaks grow higher as the value approaches the threshold. The Moving Batch Window follows the data distribution more closely than the consecutive streak, showing a clear downward trend and dropping quickly when a long sequence of solvable nodes occurs.

For the Statistical SPRT, the optimal parameters and stopping points differ a lot between the two models. In the RF case, the SPRT takes a higher risk and stops early, whereas in the QB case, it is safer and runs more computations. This is due to the optimized $\alpha$ and $\beta$ values, which are inverted between the two models. The QB model has a weaker ranking capacity compared to the Random Forest model. To prevent a premature stop based on initial noise, its Type I error threshold $\alpha$ is decreased. In contrast, the RF model provides a cleaner sorting, the SPRT can safely focus on minimizing the Type II error $\beta$ to reduce computation time.

The MAB UCB Explorer and Thompson Posterior 95% limit reflect the observed data, but are horizontally mirrored. Although the MAB path fluctuates in the first 40 steps, the Thompson metric shows a steady, smooth increase. The Thompson metric is consistently more risk-averse than the MAB. The MAB stops before a final group of spread-out unsolved DRPs, whereas the Thompson metric continues deeper into the list. The Thompson engine requires more information to ensure no further meltdowns remain, which results in zero leakage in both cases. This behavior is caused by the 95% confidence limit, which requires high certainty that no conflicted DRPs remain down the line.

Finally, the metrics change their relative stopping order between the two models. This shows that difficulty score ranking strongly influences certainty metrics. The metrics adjust well to the ranking variations and successfully react to the observed data to ensure that any ranking mistakes are handled safely.

## 9.5 Sensitivity Analysis & Robustness

To investigate the robustness of each certainty metric, the final test case scenario is used. All individual runs within this scenario are evaluated to perform a robustness sensitivity analysis. This analysis utilizes the best-performing difficulty scoring metric, which is the Random Forest model. The analysis focuses on three key metrics: the number of missed conflicted DRPs, the percentage of skipped computations, and the number of wasted solves, which are redundant compliant DRP calculations. The results are summarized in Table 9.3.

| **Certainty Metric** | **Missed Conflicted DRPs (Count)** | **Computations Skipped (%)** | **Wasted Solves (Count)** |
|---|---|---|---|
| Batch Window Solvability | 2.45 ± 6.04 | 62.97 ± 9.57 | 20.1 ± 3.4 |
| Thompson Posterior | 1.69 ± 3.44 | 61.61 ± 12.40 | 22.1 ± 12.0 |
| Consecutive Solves Streak | 1.24 ± 2.10 | 59.26 ± 10.85 | 23.1 ± 6.2 |
| MAB UCB Explorer | 2.69 ± 6.44 | 63.90 ± 9.59 | 19.6 ± 4.1 |
| Sliding Percentile | 0.93 ± 1.25 | 58.28 ± 13.84 | 25.1 ± 14.3 |
| Statistical SPRT | 2.45 ± 5.39 | 63.59 ± 9.12 | 19.8 ± 3.6 |

**Table 9.3:** Out-of-Sample Robustness Sensitivity Analysis (Mean ± Standard Deviation).

Table 9.3 shows the performance results in all the evaluated runs. For missed conflicted DRPs, the error variance varies notably between the methods. The Sliding Percentile shows the lowest average in the missed conflicts category ($0.93 \pm 1.25$) and the tightest standard deviation. In contrast, the Multi-Armed Bandit (MAB) UCB framework is the most aggressive, missing an average of 2.69 conflicted DRPs with the highest observed standard deviation (6.44). In terms of computational savings, Statistical SPRT consistently stops early, skipping an average of 63.59% of computations with the lowest standard deviation (9.12%). Metrics can be divided into two classes: risk-averse and efficiency-focused. The risk-averse group includes Thompson Posterior, Consecutive Solves Streak, and Sliding Percentile. Within this group, the Sliding Percentage is the most conservative, whereas the Consecutive Solves Streak is the most daring. The Thompson Posterior method acts as the median in all categories, representing a balanced alternative within the risk-averse group. The efficiency-focused cluster consists of Batch Window Solvability, MAB UCB, and Statistical SPRT. These approaches accept a higher constraint leakage to maximize skipped computations and reduce wasted solves. Within this group, Statistical SPRT is the most stable, maintaining the lowest standard deviation in both leakage and skipped computations while matching the low wasted solve rate of the other efficiency metrics.

## 9.6 Feature Reduction and Computational Optimization

The ranking models require several features of conflict analysis as input. To generate these variables, the current model must calculate all possible conflicts and isolate which instances are ignorable compared to the original timetable. These features are essential for applying the Station Conflict Model (SCM) at a specific DRP. Removing these conflict-based covariates from the ranking models allows the ranking approach to rank the DRPs without generating each conflict feature. Under this reduced construction, potential conflicts are only computed for each DRP as it is taken from the sorted queue for evaluation. If a stopping method terminates early, the preprocessing and building times for the remaining portion of the queue are skipped entirely.

The time to produce these features varies per iteration depending on the specific PESP modifications, but on average, the time spent on preprocessing is 6.42 times longer than the time required for the SCM to solve the DRP. Skipping these build actions offers a large computational savings. To evaluate this potential, each ranking model is reduced, removing the possible conflicts and the ignorable conflict covariate.

The results showed that the Random Forest model remained the top performer, showing only a minor decrease in performance compared to the full model (Table 9.4).

| Metric | Spearman $\rho$ | AUC | Opt. Thresh. | Acc. | FNR | Dynamic PPS |
|---|---|---|---|---|---|---|
| RF Model | **0.4386** | **0.8444** | 0.3714 | 0.7914 | 0.2022 | 953.0000 |

**Table 9.4:** Cross-Run Performance Scoreboard for the Top-Performing Reduced RF Model Variant.

Figure 9.4 illustrates the impact of the early stopping methods on total execution time. The results reveal a clear difference in time between the approaches. The metrics are again divided into two groups, yet there is a significant change; while the risk-averse group previously consisted of the Bayesian Thompson, Consecutive Solves Streak, and Sliding Percentile, the Consecutive Solves metric has switched places with the Statistical SPRT. In this reduced-feature environment, the SPRT becomes more conservative, and the Consecutive Solves has become even more daring. Nevertheless, the SPRT still saves the most time within the risk-averse group.

For maximum computational reduction, the Batch Window Solvability and the Consecutive Solves Streak provide identical performance. Both methods stop in the queue early enough to achieve a runtime reduction over 75%, closely followed by the Multi-Armed Bandit (MAB) UCB Explorer.

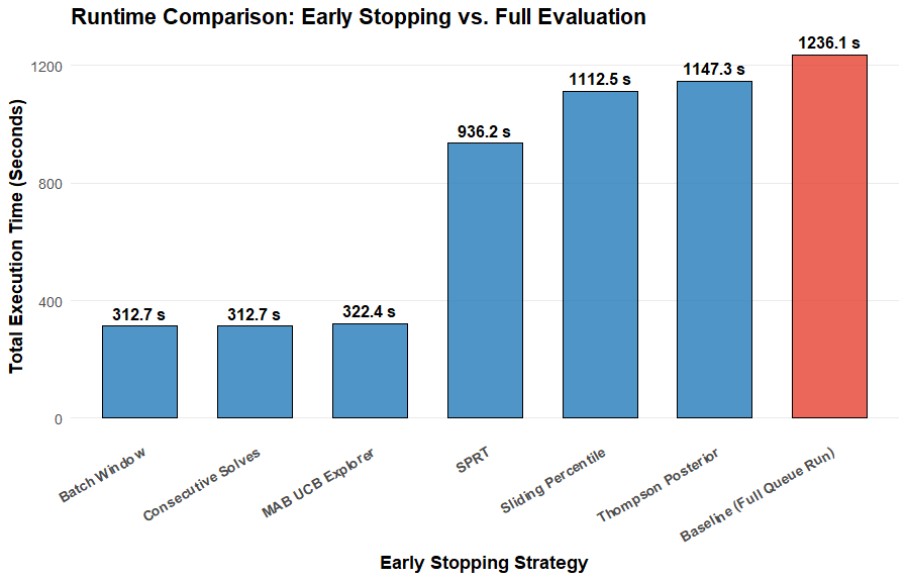


**Figure 9.4:** Total system execution time across early stopping strategies using the reduced feature set compared against the full-queue baseline reference execution.

## 9.7 Results Evaluation

The results of the experiments presented validate the advantage of the proposed "Filter, Rank and Stop" framework for rail rescheduling automation. Rather than evaluating every DRP in a massive queue, the interaction of a highly accurate difficulty ranking method with an early-stopping mechanism provides clear infrastructure benefits.

Deploying the efficiency-focused stream, such as the Random Forest paired with the Statistical SPRT, maximizes processing speed. This configuration successfully eliminates up to 63.59% of redundant calculations while keeping low predictable overhead bounds.

Conversely, when the preference is towards a more risk-averse combination, such as the Random Forest paired with the Thompson Posterior, the framework minimizes missed constraints to a negligible margin of $1.69 \pm 3.44$, while still saving on average 61.261% of processing capacity.

Furthermore, implementing the feature-reduced ranking model offers a significant second optimization technique. By moving the building of conflict features out of the initial ranking phase and building them only when the SCM is executed, the preprocessing and building time for skipped DRPs is eliminated. Given that the build time is a large part of the SCM calculation, accepting a slight reduction in the model ranking performance allows even more reduction in the total system execution time, with overall time savings exceeding 75%.

Ultimately, combining the historical risk profile with the ECDF transformation connects past station vulnerabilities directly to active machine learning models. Instead of using brute force to check every scenario, this dual-layer system allows rail operators to safely stop computations early without sacrificing network reliability.

# 10

# Conclusion

Creating an alternative train schedule is a highly complex task that requires balancing numerous variables and operational constraints to produce a feasible, optimized, and robust timetable. In collaboration with the RAAD department at ProRail, Lynxx has spent more than two years developing a two-phase model to take on this challenge. However, the current model struggles to meet ProRail's operational demands because its overall computation time remains too high. The framework consists of two layers: the macroscopic PESP model manages network changes like train path shifts and reroutes, affecting various *Dienstregelpunten* (DRPs). The microscopic SCM model then assesses each modified DRP for feasibility, identifies conflicts, and reassigns paths to find a conflict-free solution. If an SCM identifies an unsolvable DRP, it sends constraints back to the PESP model, leading to a new iteration. This process continues until both levels are feasible. Data exploration reveals that only 10 to 30% of the generated DRPs are actually unsolvable during these iterations, which highlights a huge opportunity for optimization.

This research explicitly addresses this problem by focusing on the communication between the macro-and micro levels to reduce the total computation time. Specifically, this study answers the research question of how to optimize the interaction between these layers by minimizing the number of redundant micro computations. To achieve this, a three-step Filter, Rank, and Stop method is proposed. The initial filtering stage eliminates all DRPs that contain zero potential conflicts, as it is mathematically impossible for them to be unsolvable. This simple baseline step successfully reduces the total SCM processing load by 10 percent.

The core of this research evaluates the Rank and Stop mechanism, which aims to predict which DRPs are unsolvable so that the system only uses the computing resources to the nodes that generate PESP constraints. Statistical testing during data exploration

confirmed that the actual conflict distribution is zero-inflated and comes close to a Poisson distribution. Seven difficulty ranking models were evaluated, ranging from basic linear baselines to advanced machine learning models. In all frameworks, the historical risk profile emerged as a highly significant predictor that improved model performance. Ultimately, the *Random Forest model* and the *Quasibinomial GLM* achieved the highest ranking accuracy and were selected for integration with the early-stopping mechanisms.

To establish the optimal stopping point, six distinct certainty metrics were evaluated, including static, percentile, batch-based, and statistical frameworks. Particular focus was placed on adaptive reinforcement learning and Bayesian approaches, specifically the *Multi-Armed Bandit Upper Confidence Bound (MAB UCB) Explorer* and the *Thompson Posterior 95% Limit*. The graph of the convergence paths showed that the Random Forest model is the superior difficulty ranking method, as it creates a much cleaner ranking that allows the early-stopping metrics to execute efficiently.

Consequently, the Random Forest model was used for robustness sensitivity analysis across independent runs. The sensitivity results confirmed a clear division among the certainty metrics, either more risk-averse or more computationally efficient. The optimal deployment depends entirely on organizational preferences regarding the trade-off between computational speed and missed constraints. If the priority is risk aversion, the Thompson Posterior is the ideal choice. It delivers highly stable performance under varying timetable conditions and achieves clean stop with no missing conflicted DRPs. Conversely, if the primary objective is to maximize computational savings, the *Statistical SPRT* is the better choice. Although it does miss some conflicted DRPs, it reduces up to 60% of the total processing queue and wastes on average 19 calculations on redundant solves.

Beyond optimizing the early-stopping parameters, this research evaluated another approach to reduce the runtime even more. By removing the conflict features in the ranking models, the building of conflicted DRPs is only carried out when they are actually computed by the micro algorithm. As a result the model only builds when necessary and skips building time for each DRP that is easily solvable. While this method does reduce the performance of the Random Forest Model slightly, the gains in time efficiency are substantial. When integrated with the *Batch Window* or Consecutive Solves, this approach reduces the overall time by 75%, making it the most effective strategy to reduce computation time.

This research demonstrates that when addressing the immense complexity of alternative timetabling, the opportunity for optimization lies in bridging the communication gap between the macro and micro scheduling layers. By establishing an enhanced communication channel between the two layers, the proposed Filter, Rank and Stop method transforms

the micro-computation process into a localized search. Instead of blindly routing and adjusting stations where there is no operational conflict, the algorithm traces the exact trajectory of a disruption through the network. This allows the system to follow the path of a conflict dynamically, resolving dependencies sequentially as they unfold. This level of computational efficiency sets a new benchmark for automated rescheduling, allowing large maintenance cases to be planned with little to no impact on the passenger's journey.

# 11

# Discussion and Further Research

The results of this research demonstrate that the Filter, Rank, and Stop method can significantly reduce the total computation time required for alternative train scheduling. However, several opportunities remain to refine the methodology and extend its performance in future applications.

A primary direction for future research involves the integration of topological dependencies directly into difficulty scoring metrics. Currently, the framework evaluates each DRP independently. In a real-world rail network, the stations are highly connected, which means that a network modification at one location naturally creates a ripple effect at adjacent stations due to shared tracks (1, 4). Future models could group and predict multiple stations simultaneously based on their topological distance and connectivity. Taking into account the location of the planned maintenance, the difficulty metric could automatically assign a higher risk to DRPs that are directly affected or spatially close to the disruption, as these areas are more likely to experience capacity conflicts.

Furthermore, expanding the data set with additional maintenance scenarios would likely improve the accuracy of the model. A larger and more diverse training set would strengthen the pattern detection capabilities of the Random Forest model and provide a more detailed background for the historical asset risk profile. In addition, future experiments should test the framework under complex conditions in which multiple maintenance tasks occur simultaneously. Investigating these combined cases would reveal whether early-stopping methods can successfully predict different conflicts based on insights gained from single-maintenance scenarios.

Finally, the multi-iteration behavior of the scheduling process offers a clear path for optimization. A current limitation of this study is that the framework was evaluated in

only three macro-micro cycles. Future work should analyze longer iteration runs to observe how well the stopping metrics scale over several iterations.

Rather than treating each iteration as an independent event, future implementations can exploit the correlation of unsolvability across consecutive cycles to accelerate convergence. If a specific DRP node consistently generates unsolvable states and returns constraints in earlier iterations, its difficulty weight can be dynamically increased in the next queue. In contrast, nodes that consistently demonstrate high flexibility and remain conflict-free can be further pushed down the ranking. Leveraging this inter-iteration history would allow the model to bypass re-evaluating recurring problems, creating an even more adaptive communication loop between the macro and micro scheduling layers.

# References


[1] Gert-Jaap Polinder. *Resolving infeasibilities in the PESP model of the Dutch railway timetabling problem.* PhD thesis, August 2015. 1, 5, 7, 8, 9, 73

[2] Paolo Serafini and Walter Ukovich. **A Mathematical Model for Periodic Scheduling Problems**. *SIAM Journal on Discrete Mathematics*, pages 550–581, August 2006. 1, 10, 12, 14

[3] Jean-François Cordeau, Paolo Toth, and Daniele Vigo. **A Survey of Optimization Models for Train Routing and Scheduling**. *Transportation Science*, **32**(4):380–404, November 1998. 1, 28

[4] Michael Kümmling, Peter Grossmann, Karl Nachtigall, Jens Opitz, and Reyk Weiss. **A state-of-the-art realization of cyclic railway timetable computation**. *Public Transport*, **7**(3):281–293, December 2015. 1, 7, 8, 73

[5] Antonio Lova, María Tormos, Federico Barber, Laura Ingolotti, Miguel Salido, and Montserrat Abril. **Intelligent Train Scheduling on a High-Loaded Railway Network.** pages 219–232, January 2004. 1, 7, 10, 14

[6] Prorail. **Zo plannen we werkzaamheden**. 2

[7] Leo Kroon, Dennis Huisman, Erwin Abbink, Pieter-Jan Fioole, Matteo Fischetti, Gábor Maróti, Alexander Schrijver, Adri Steenbeek, and Roelof Ybema. **The New Dutch Timetable: The OR Revolution**. *Interfaces*, **39**(1):6–17, February 2009. 5, 14, 20

[8] MingLi Zhao, ShaoQuan Ni, ZhiGang Du, XiaoLong Ma, and YangYang Liu. **Train Scheduling in High-Speed Railway Freight Transportation Using a Hyper-Heuristic Algorithm**. *Transportation Research Record: Journal of the Transportation Research Board*, **2679**, December 2024. 5, 14

[9] Aron van Woerkom. **Into the Business - Lynxx**, April 2019. 6

[10] Sander Van Aken, Nikola Bešinović, and Rob M. P. Goverde. **Designing alternative railway timetables under infrastructure maintenance possessions**. *Transportation Research Part B: Methodological*, **98**:224–238, April 2017. 6, 8, 12, 14, 19, 28

[11] Stefan Binder, Yousef Maknoon, and Michel Bierlaire. **The multi-objective railway timetable rescheduling problem**. *Transportation Research Part C: Emerging Technologies*, **78**:78–94, May 2017. 6, 10, 13, 14

[12] Anita Schöbel. **An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation**. *Transportation Research Part C: Emerging Technologies*, **74**:348–365, January 2017. 6, 13

[13] **ProRail Jaarverslag 2025**. 2025. 6

[14] **RNE – RailNetEurope | Association For Facilitating Traffic On European Rail Infrastructure**. 6

[15] Florin Leutwiler and Francesco Corman. **A logic-based Benders decomposition for microscopic railway timetable planning**. *European Journal of Operational Research*, **303**(2):525–540, December 2022. 7, 10, 12, 13, 22, 28

[16] Thomas Schlechte, Ralf Borndörfer, Berkan Erol, Thomas Graffagnino, and Elmar Swarat. **Micro–macro transformation of railway networks**. *Journal of Rail Transport Planning & Management*, **1**(1):38–48, November 2011. 7, 12

[17] Mingxuan Zhong, Yixiang Yue, Leishan Zhou, and Jianping Zhu. **Parallel optimization method of train scheduling and shunting at complex high-speed railway stations**. *Computer-Aided Civil and Infrastructure Engineering*, **39**(5):731–755, 2024. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/mice.13077. 7, 43

[18] Xingyu Zhou. **AI-Powered Two-Phase Method for Microscopic Periodic Railway Operation Diagrams**. *INNO-PRESS: Journal of Emerging Applied AI*, **1**(4), July 2025. 7, 11

[19] Nikola Bešinović, Rob M. P. Goverde, Egidio Quaglietta, and Roberto Roberti. **An integrated micro–macro approach to robust railway timetabling**. *Transportation Research Part B: Methodological*, **87**:14–32, May 2016. 7, 12, 14

[20] Sabrina Herrigel, Marco Laumanns, Jacint Szabo, and Ulrich Weidmann. **Periodic railway timetabling with sequential decomposition in the PESP model**. *Journal of Rail Transport Planning & Management*, **8**(3):167–183, December 2018. 7, 10

[21] Christian Liebchen and Rolf Möhring. **The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond**. In *Lecture Notes in Economics and Mathematical Systems*, **600**, pages 117–150. January 2008. Journal Abbreviation: Lecture Notes in Economics and Mathematical Systems. 9, 14, 19

[22] Shi Qiang Liu and Erhan Kozan. **Scheduling trains as a blocking parallel-machine job shop scheduling problem**. *Computers & Operations Research*, **36**(10):2840–2852, October 2009. 9, 28

[23] Ahmad Reza Jafarian-Moghaddam. **Elastic train scheduling model**. *Applied Soft Computing*, **110**:107627, October 2021. 9

[24] Florian Fuchs, Bernardo Martin-Iradi, and Francesco Corman. **Optimizing periodic stability in railway timetables: A microscopic model for networks with a macroscopic comparison**. *Journal of Rail Transport Planning & Management*, **36**:100555, December 2025. 9, 33

[25] Enrico Bortoletto, Rolf van Lieshout, Berenike Masing, and Niels Lindner. **Periodic Event Scheduling with Flexible Infrastructure Assignment**. *24th Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2024)*, pages 4:1–4:18, October 2024. 9, 13, 22, 28

[26] David S. Johnson. **Review of Combinatorial Optimization: Algorithms and Complexity**. *The American Mathematical Monthly*, **91**(3):209–211, 1984. 10

[27] Michiel A. Odijk. **A constraint generation algorithm for the construction of periodic railway timetables**. *Transportation Research Part B: Methodological*, **30**(6):455–464, December 1996. 10, 28

[28] Marc Goerigk. **Exact and heuristic approaches to the robust periodic event scheduling problem**. *Public Transport*, **7**(1):101–119, March 2015. 11

[29] Qing Wu, Maksym Spiryagin, Colin Cole, and Tim McSweeney. **Parallel computing in railway research**. *International Journal of Rail Transportation*, **8**(2):111–134, April 2020. _eprint: https://doi.org/10.1080/23248378.2018.1553115. 11

[30] Krisanarach Nitisiri, Mitsuo Gen, and Hayato Ohwada. **A parallel multi-objective genetic algorithm with learning based mutation for railway scheduling**. *Computers & Industrial Engineering*, **130**:381–394, April 2019. 11

[31] Montserrat Abril, Miguel A. Salido, and Federico Barber. **Distributed search in railway scheduling problems**. *Engineering Applications of Artificial Intelligence*, **21**(5):744–755, August 2008. 11

[32] Gert-Jaap Polinder, Thomas Breugem, Twan Dollevoet, and Gábor Maróti. **An adjustable robust optimization approach for periodic timetabling**. *Transportation Research Part B: Methodological*, **128**:50–68, October 2019. 14

[33] Valentina Cacchiani, Dennis Huisman, Martin Kidd, Leo Kroon, Paolo Toth, Lucas Veelenturf, and Joris Wagenaar. **An overview of recovery models and algorithms for real-time railway rescheduling**. *Transportation Research Part B: Methodological*, **63**:15–37, May 2014. 20, 30

[34] Ilse Louwerse and Dennis Huisman. **Adjusting a railway timetable in case of partial or complete blockades**. *European Journal of Operational Research*, **235**(3):583–593, June 2014. 29, 30

[35] Lucas P. Veelenturf, Martin P. Kidd, Valentina Cacchiani, Leo G. Kroon, and Paolo Toth. **A Railway Timetable Rescheduling Approach for Handling Large-Scale Disruptions**. *Transportation Science*, **50**(3):841–862, August 2016. 29, 30

[36] Peter Eklund, Stephen Kirkby, and S. Pollitt. **A dynamic multi-source Dijkstra's algorithm for vehicle routing**. pages 329–333, December 1996. 32

[37] Xinyan Zhang, Himel Mallick, Zaixiang Tang, Lei Zhang, Xiangqin Cui, Andrew K. Benson, and Nengjun Yi. **Negative binomial mixed models for analyzing microbiome count data**. *BMC Bioinformatics*, **18**(1):4, January 2017. 44

[38] R. W. M. Wedderburn. **Quasi-Likelihood Functions, Generalized Linear Models, and the Gauss-Newton Method**. *Biometrika*, **61**(3):439–447, 1974. 45

[39] Roger Koenker and Kevin F. Hallock. **Quantile Regression**. *Journal of Economic Perspectives*, **15**(4):143–156, December 2001. 46

[40] Leo Breiman. **Random Forests**. *Machine Learning*, **45**(1):5–32, October 2001. 47

[41] Tianqi Chen and Carlos Guestrin. **XGBoost: A Scalable Tree Boosting System**. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, August 2016. arXiv:1603.02754 [cs.LG]. 48

[42] A. Wald. **Sequential Tests of Statistical Hypotheses**. *The Annals of Mathematical Statistics*, **16**(2):117–186, June 1945. 50

[43] R.S. Sutton and A.G. Barto. **Reinforcement Learning: An Introduction**. *IEEE Transactions on Neural Networks*, **9**(5):1054–1054, September 1998. 51

[44] Herbert Robbins. **Some aspects of the sequential design of experiments**. *Bulletin of the American Mathematical Society*, **58**(5):527–535, 1952. 51

[45] Michael N. Katehakis and Arthur F. Veinott. **The Multi-Armed Bandit Problem: Decomposition and Computation**. *Mathematics of Operations Research*, **12**(2):262–268, 1987. 52

[46] William R. Thompson. **On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples**. *Biometrika*, **25**(3/4):285–294, 1933. 52

[47] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. **A Tutorial on Thompson Sampling**, July 2020. arXiv:1707.02038 [cs.LG]. 52

# Appendix

**Table 11.1:** Mathematical Model Framework Configurations with Structural Risk Integration

| **Difficulty Metric** | **Equation Formulation** |
|---|---|
| Linear Baseline | $y = \beta_0 + \beta_1 \text{norm\_pos\_conflicts} + \beta_2 \text{norm\_ign\_conflicts} + \beta_3 \text{norm\_busy} + \beta_4 \text{norm\_keringen} + \beta_5 \text{norm\_tijdsverleggingen} + \beta_6 \text{norm\_omleidingen} + \beta_7 \text{norm\_changes} + \beta_8 \text{norm\_no\_trains} + \delta \text{drp\_historical\_risk} + \epsilon$ |
| Negative Binomial GLMM | $\log(\mu) = \beta_0 + \beta_1(\text{norm\_pos\_conflicts} \times \text{size\_cat}) + \beta_2(\text{norm\_changes} \times \text{size\_cat}) + \beta_3 \text{norm\_keringen} + \beta_4 \text{norm\_tijdsverleggingen} + \beta_5 \text{norm\_omleidingen} + \beta_6 \text{norm\_no\_trains} + \delta \text{drp\_historical\_risk} + u_{\text{drp}}$ |
| Quasi-Poisson GLMM | $\log(\mu) = \beta_0 + \beta_1(\text{norm\_pos\_conflicts} \times \text{size\_cat}) + \beta_2(\text{norm\_changes} \times \text{size\_cat}) + \beta_3 \text{norm\_ign\_conflicts} + \beta_4 \text{norm\_keringen} + \beta_5 \text{norm\_tijdsverleggingen} + \beta_6 \text{norm\_omleidingen} + \delta \text{drp\_historical\_risk} + u_{\text{drp}}$ |
| Quasibinomial Bounded GLM | $\text{logit}(\tilde{y}) = \beta_0 + \beta_1(\text{norm\_pos\_conflicts} \times \text{size\_cat}) + \beta_2(\text{norm\_changes} \times \text{size\_cat}) + \beta_3 \text{norm\_keringen} + \beta_4 \text{norm\_tijdsverleggingen} + \beta_5 \text{norm\_no\_trains} + \beta_6 \text{norm\_busy} + \delta \text{drp\_historical\_risk}$ |
| 90th Percentile Quantile | $Q_{0.90}(y \mid \mathbf{X}) = \beta_0^{(0.90)} + \beta_1^{(0.90)}(\text{norm\_pos\_conflicts} \times \text{size\_cat}) + \beta_2^{(0.90)}(\text{norm\_changes} \times \text{size\_cat}) + \beta_3^{(0.90)} \text{norm\_ign\_conflicts} + \beta_4^{(0.90)} \text{norm\_keringen} + \beta_5^{(0.90)} \text{norm\_omleidingen} + \beta_6^{(0.90)} \text{norm\_tijdsverleggingen} + \beta_7^{(0.90)} \text{norm\_no\_trains} + \beta_8^{(0.90)} \text{norm\_busy} + \delta^{(0.90)} \text{drp\_historical\_risk}$ |
| Random Forest Engine | $y = f_{\text{RandomForest}}(\text{norm\_pos\_conflicts}, \text{norm\_ign\_conflicts}, \text{norm\_busy}, \text{norm\_keringen}, \text{norm\_tijdsverleggingen}, \text{norm\_changes}, \text{drp\_historical\_risk})$ |
| Gradient Boosting Engine | $y = g_{\text{XGBoost}}(\text{norm\_pos\_conflicts}, \text{norm\_ign\_conflicts}, \text{norm\_busy}, \text{norm\_keringen}, \text{norm\_tijdsverleggingen}, \text{norm\_changes}, \text{norm\_no\_trains}, \text{drp\_historical\_risk})$ |