Creating Robust Train Unit Shunting Plans using Probabilistic Programming

G.B.J. van Zwienen[1]
Supervisors: S. Dumančić[1]**, I.K. Hanou**[1]**, R.J. Gardos Reid**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 21, 2025

Student: G.B.J. van Zwienen
Final project course: CSE3000 Research Project
Thesis committee: S. Dumančić, I.K. Hanou, R.J. Gardos Reid, N.M. Gürel

Abstract

Train Unit Shunting is a complex process that directs trains through a shunting yard. In real-world railway operations, disturbances are common, requiring shunting schedules to be robust against uncertainties such as delays. Previous research has proposed algorithms for the Train Unit Shunting Problem (TUSP) and one study attempted to create robust shunting plans by defining a probabilistic model of the uncertainties involved and inferring a distribution of robust solutions for the TUSP. Following this approach, this paper investigates the use of probabilistic programming in increasing robustness of shunting plans using an advanced TUSP solver. This research develops a model for uncertain shunting scenarios, solves these scenarios, and applies importance sampling to infer the posterior distribution, producing a distribution of robust shunting plans instead of a single plan. The paper presents examples demonstrating that it is beneficial to use one of the output robust plans over the plan made for the deterministic scenario, revealing the potential of integrating probabilistic programming techniques into the planning process to improve railway efficiency and reduce delays.

1 Introduction

Planning algorithms play a crucial role in modern systems and are widely used, from coordinating material distribution in factories to managing traffic at airports, in networks, and across train stations. While these algorithms are powerful tools, applying them to real-world scenarios comes with additional challenges. In practice, numerous uncertainties must be accounted for. Some materials in the factory might be out of stock, a machine suddenly goes out of order, or a train can have delay. The plans an algorithm makes must remain feasible even after disruptions occur. In other words, they must be robust against uncertainties.

This research will focus on the Train Unit Shunting Problem (TUSP), originally introduced by Freling et al. [9]. The TUSP involves directing trains through a shunting yard, where they are parked, serviced, and cleaned during periods of inactivity. Each train must leave the shunting yard in a specific order and at the scheduled time. Because the problem deals with passenger trains in the real world, various uncertainties such as delays and technical issues must be considered. It is therefore important that the plans an algorithm makes remain effective, or at least do not get too disrupted when unexpected events occur.

Various algorithms have been designed for solving the TUSP. Van der Linden et al. have developed both an algorithm and a simulator for the TUSP [24], and several attempts have been made at making more robust algorithms, for example by R.W. van den Broek [23]. R.J. Gardos Reid proposed an interesting approach on improving robustness of algorithms, using probabilistic programming techniques to create a generative model of the TUSP. The core idea is that a probabilistic model is made for the input, capturing the uncertainties. Samples from this model are subsequently fed into an existing algorithm for the TUSP. By applying probabilistic inference on the resulting plans, the result is a distribution of robust solutions [18]. One key advantage of this approach is that this method can be applied on any planning problem which has an existing algorithm, even without having a deep understanding of how the algorithm works.

This research uses probabilistic programming to create robust TUSP plans. It builds on the work of Gardos Reid by introducing a new form of uncertainty and applying the method

on a more advanced solver. Additionally, it proposes a way to apply the resulting output distribution in practice.

This motivates the following research question: *'How can one model uncertainties in the Train Unit Shunting Problem and use this to create a robust version of an existing planning algorithm?'*. The main question can be broken down into the following sub-questions:

1. *What uncertainty might exist in a real-world version of the train unit shunting problem?*
2. *How can uncertainties in the train unit shunting problem be modelled as a probabilistic program?*
3. *Which inference techniques can be applied to the resulting probabilistic program and how well do they perform?*
4. *How can the output distribution of the plans be used to plan in the uncertain version of the train unit shunting problem?*

The next section will explain the TUSP in detail, dive deeper into the background of probabilistic programming, and show related work in this area of research. Section 3 explains the methods used to answer the research questions, after which section 4 shows the setup and results of the experiments. Section 5 reflects on the ethical aspects of the research and discusses the reproducibility of the methods. The paper will conclude with a discussion of the results in section 6 and the conclusion and directions for future research in section 7.

2 Background and Related work

2.1 The Train Unit Shunting Problem

Before describing the problem, the terminology used must be explained. A *train* is the entire vehicle used to transport, for example, passengers. A train can consist of multiple *train units*. Examples of train units include an ICM-3 or an SLT-4, where the letters indicate the type of equipment and the number refers to the number of carriages in the train unit. Train units of the same equipment type can be coupled to form a train. For example, an ICM-3 train unit can be combined with an ICM-4 to create a train consisting of seven carriages in total. In the TUSP, train units are not split further into carriages, so train units are the smallest entity.

Trains are not always used. Especially outside of rush hour, not all available trains are needed. Trains which are not in use can be parked at a shunting yard, which is a large collection of tracks and can be seen as a parking place for trains. The *shunting* of train units has three components: routing, matching, and parking [23]. Trains are routed through the tracks of the shunting yard, and can be split into train units. These units can be combined again into new trains. Finally, a train unit can be parked on a normal track or at specialised facilities where it can be serviced or cleaned.

The Train Unit Shunting Problem (TUSP) deals with the process of navigating trains over a shunting yard and can quickly become complicated when considering space/time limits, servicing tasks done on limited amount of servicing space or personnel, and trains already

located on the yards. Freling et al. showed that the basic TUSP is NP-hard [9]. In fact, the separate components of only matching, parking or routing trains are already NP-hard problems [21] and a complex problem arises when combining these as well as considering service tasks. Figure 1 shows a basic shunting example where the train units of three arriving trains are recombined into two departing trains.

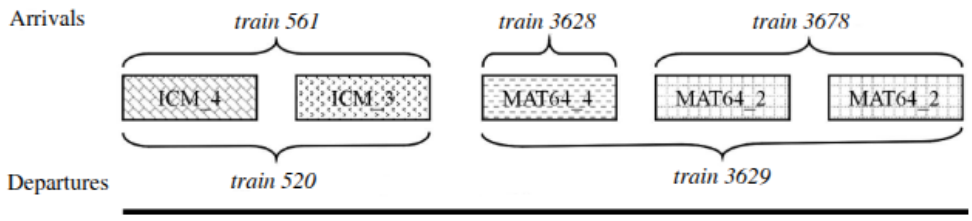

Figure 1: Shunting Example by Freling et al. [9]. Three arriving trains are recombined into two departing trains.

On average, 11 disruptions are reported every single day in railway transportation in The Netherlands [6]. There are many different causes for disruptions on the rails but they can all be categorised into three groups: the weather, people, and technology. Railway disruptions almost always cause the journey of passengers to be delayed [3]. Given the many uncertainties in train traffic, it is crucial that the final shunting plan is robust to sudden changes. Ideally, the plan should still be executable even if a train arrives slightly later than scheduled or if any equipment failures occur. To evaluate whether a plan is robust, a definition of *robustness* is required. The method developed here is initially applied to simple scenarios in order to ease analysis, which also motivates the use of a simple definition of robustness. The departure delay of a train is the difference between the scheduled departure time and the actual departure time. The robustness of a plan will be measured by taking the sum of the departure delays of all outgoing trains from the shunting yard. A plan is considered robust when this value is close to or equal to zero, indicating minimal deviation from the original schedule.

2.2 Probabilistic Programming

This research explores the application of probabilistic programming to generate robust train unit shunting plans. Probabilistic programs extend traditional programs by two key components: they are capable of drawing random variables from distributions, and the programs can use known data to guide these random choices [13]. A probabilistic program defines a model for quantities that are measurable in the real world. The values produced by the model can be compared to the actual values observed in real-world data, which are called the *observations* [20]. This research uses a probabilistic programming language as 'they provide direct support for defining probabilistic models and performing probabilistic inference within the language' [17, p. 603]. An example of a probabilistic programming framework is Gen.jl. The Gen framework for the Julia programming language is a general purpose probabilistic programming system, designed to achieve high modelling flexibility and inference efficiency. Gen is introduced by Cusumano Towner et al. in 2019 and outperforms state-of-the-art probabilistic programming systems [8]. Gen provides a library of built-in probability distributions, and the inference library of Gen is useful for writing efficient probabilistic

inference algorithms [4].

2.3 Probabilistic Inference

Once a model and observations are specified, inference methods can be applied to recover the posterior distribution of unobserved variables in the model. Given the observed data, the goal is to infer the underlying causes, that is to estimate the distribution from which these samples were generated [12, Chapter 3].

Several different inference algorithms have been developed to approximate posterior distributions. Among these, Monte Carlo (MC) methods operate by generating a large collection of samples from a model to represent the conditional distribution. Rejection sampling is a MC inference method which works by continuously sampling from the entire model, but only keeping the samples which adhere to a certain condition. Rejection sampling is a simple and useful method but it is often not efficient. Even if there are samples that exactly observe the condition, it can take a large number of samples to find them. A slightly different approach is Importance Sampling. This method keeps all samples, but gives each sample a weight based on the condition. This way it is able to approximate the target distribution by weighing the random draws from a different distribution [19].

The method of independent sampling works in simple situations, but can become difficult or impossible when dealing with complex distributions [11]. Markov chain Monte Carlo (MCMC) takes the approach where instead of taking random independent samples from a model, it iterates on a single sample. A Markov chain is a system consisting of a set of states and transition functions between those states. From every state there are certain probabilities of proceeding to different states. MCMC forms a Markov chain by taking a single sample from an initial distribution, and iterating on this sample by introducing small changes on each iteration [12, Chapter 8].

2.4 Related work

The researcher R.W. van den Broek has conducted multiple studies to train unit shunting algorithms. In his Master's thesis in 2016, he presented a heuristic which improved the creation of shunt plans when compared to methods used before [21]. This research was done at the request of NedTrain, a subsidiary of the Nederlandse Spoorwegen (Dutch Railways, NS). This local search based algorithm assumed that both the arrival times and the service task durations are deterministic, which is often not the case in practice. Extending on these ideas, Van den Broek later works on an algorithm for the train unit shunting problem with service scheduling (TUSPwSS) in a paper in 2021 [22] and his thesis in 2022 [23]. In addition to the classical TUSP, this algorithm is also able to schedule different kinds of services for train units. In his study in 2022, Van den Broek researched robust planning as well, by including 'identification of strong predictors of the robustness of shunting plans to small disturbances' [23, p. 8]. This research only considered the train arrival times and the processing times of tasks as uncertain variables. More research has been done on making a TUSP algorithm robust by Kleine [14], who proposed a non-deterministic simulated annealing method, taking into account arrival time and task duration uncertainties during scheduling. The research built upon the algorithm of Van den Broek [21] as it existed on March 1, 2019, by guiding the simulated annealing process towards robust solutions. The

proposed algorithm significantly increases robustness of the generated shunting plans, but the robustness is directly integrated into the Simulated Annealing function, which makes this approach not flexible towards other uncertainties in the TUSP.

While the above approaches modify or create an algorithm in order to achieve robustness, an alternative method has been proposed by R. J. Gardos Reid [18]. He attempted to create robust plans using an existing TUSP algorithm without modifications. An algorithm can be used as a 'black box', and the logic for increasing robustness is built around this. An advantage of this approach is that it can be applied to any problem assuming that there exists a planner for the problem. Using this method, a robust planning algorithm can be made, even without knowing the inner workings of such a planner in detail. In order to implement uncertainties, Gardos Reid utilised probabilistic programming to make a probabilistic model of the input. He then sampled from this model to obtain randomised scenarios and executed the algorithm. After applying inference methods on the resulting plans, this results in a distribution of robust plans/solutions to the problem. The research found that for simple scenarios, the robust plans had a higher success rate than the plans produced directly by the solver, but the performance of the solver that was used was insufficient to handle more complex scenarios.

To facilitate the evaluation of TUSP algorithms, Van der Linden et al. introduced an event-based simulator for the TUSP called TORS (Dutch acronym for Train Shunting and Servicing Simulator) in a paper in 2021 [24]. This tool keeps track of the state that the shunting yards and the trains are in during the shunting process and can help the evaluation of planning outcomes. Together with a shunting scenario generator and the solver developed by Van den Broek, these tools have been combined into a platform[1] maintained by the research group Robust Rail[2].

Perhaps the most frequent and impactful uncertainties in train operations is delay. A thesis by A. Krudde in 2024 contains an extensive analysis of arrival time distributions and delay of trains in shunting yards. GPS data of trains moving around in the Netherlands for over 10 months has been used to find monthly patterns. The analysis shows that in the months May till August, most trains arrive with less than one minute delay. In the following months however, many trains are delayed by 3 minutes. In September, many trains even exceed 4 minutes of delay [15].

3 Method

This research builds upon the work done by Gardos Reid [18] by following his proposed method of using probabilistic programming to generate robust plans for the TUSP. First, an uncertainty in train shunting is selected and modelled probabilistically. The resulting model is subsequently sampled to generate a set of scenarios, which are provided as an input to an existing planning algorithm. This process produces a collection of many possible plans, which together represent the prior distribution over the planning space. Lastly, in order to obtain the posterior distribution of robust plans, probabilistic inference is applied on the output plans. The general structure of the method is shown in figure 2 and each component

[1] https://github.com/Robust-Rail-NL
[2] https://www.tudelft.nl/delftrail/projects/robust-rail

is discussed in detail in the subsections below.

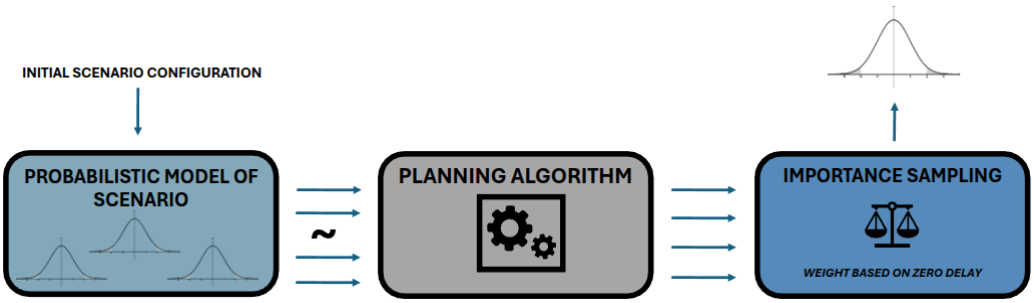

Figure 2: Creating robust solutions: A scenario configuration file is given as input and is modelled with an uncertainty. Samples from this model are used as input to a planner. Importance sampling is applied on the resulting plans to infer the final posterior distribution of plans.

3.1 Modelling uncertainty

The first step in the process is to construct a probabilistic model of the scenarios that incorporates uncertainties. This research used the framework Gen.jl to define this model. The uncertainty that will be modelled is the arrival times: trains can have a small amount of delay or arrive slightly earlier than planned. This variability is modelled by representing arrival times as stochastic variables. The train arrival times are defined in the scenario configuration file and are modelled by normal distributions centred around the original values. New arrival times are sampled from these distributions, resulting in a modified configuration where the arrival times are random.

A generative model can incorporate various other uncertainties beyond arrival times. One additional example explored in this paper is randomizing the time required to clean a train carriage. This requires a shunting yard with a cleaning track and a scenario in which a train is scheduled for cleaning. The cleaning duration can now be sampled from a normal distribution (or any other distribution) in a similar way as the arrival times. Furthermore, infrastructure failure can be modelled by changing the shunting location rather than the scenario. For example, a non-operational switch can be modelled by removing it from the location, or the length of a track can be shortened to simulate a partial closure. This research however has purely focussed on uncertain scenarios and therefore left the shunting locations unchanged.

3.2 Planning algorithm and project setup

To be able to create shunting plans, the aforementioned TUSP tools of Robust Rail were used. This includes the solver: the black box where the probabilistic model is 'wrapped around'. Additionally, the scenario generator was used as a helper tool to generate different shunting scenarios.

The solver requires two inputs: a location and a scenario. The location is a description

of a shunting yard and can be created by adding track parts like switches, bumpers and railroads. Furthermore, service facilities can be added to railroads, including cleaning platforms, washing machines and inspection pits. The scenarios contain a list of incoming and outgoing trains. For every train, scenarios describe the arrival and departure times/tracks, information about the type of train units, and potential servicing that is required. A scenario is generated by using the generator of Robust Rail, which takes a scenario configuration file as input. The configuration file specifies the servicing tasks, the train units involved, and the set of incoming and outgoing trains including their scheduled times and assigned tracks.

Once a scenario and location are defined, they are provided as an input to the solver, which computes a feasible output plan for the TUSP. Listing 1 shows a simplified example of a generative function written in Gen where arrival times are sampled randomly, a plan is made, and the delays which are required for the inference are calculated. The *@trace* statement in Gen records a random choice made during execution so it can be tracked and used later for inference.

```
@gen function generate_plan(scenario_configuration, location)
    # Sample from the model
    sampled_scenario_configuration = @trace(sample_arrival_times(scenario_configuration))

    # Create a shunt plan
    scenario = run_generator(sampled_scenario_configuration)
    departure_times, plan = run_solver(scenario, location)

    # Calculate the delays
    delays = @trace(calculate_delays(scenario_configuration, departure_times))
    return plan
end
```

Listing 1: An example of a generative function for shunting plans written in Gen.

The solver uses Tabu Search and Simulated Annealing techniques to iteratively improve solutions. The original authors explain simulated annealing in the solver as follows: 'A simulated annealing algorithm randomly selects a neighbour and accepts it immediately as the candidate solution for the next iteration if it is an improvement over the current solution. If the selected solution is worse, it is accepted with a certain probability depending on the difference in solution quality and the state of the search process.' [22, p. 149]. In practice, this means that the solver begins with an initial shunting plan, after which a small change is made at random to get a slightly different plan (neighbour). Whenever this new plan is better than the initial plan, it is accepted only with a certain probability. As a result, on different runs of the solver, different neighbours can be proposed and accepted in this process, making the search process non-deterministic. As the method of this paper is building around the algorithm it is not required to know further details of these techniques, but it is noteworthy that they make use of randomness. This is an important consideration when interpreting the results in section 4.

3.3 Inference

Now that a generative model for uncertain scenario configurations has been created and the solver has been set up, the prior distribution of plans can be constructed. The generative model is sampled to obtain scenario configuration files, which are given as an input to the

scenario generator. The generator produces scenarios, which can together with a location be fed into the solver of Robust Rail, resulting in a collection of plans: the prior distribution.

After constructing the prior distribution of plans, the final step is to obtain the posterior distribution of robust plans by applying probabilistic inference. In this study, importance sampling is applied, which is directly supported by Gen. The sampling is guided by the delay of the outgoing trains in the plan, which is computed by subtracting the original scheduled departure time from the actual departure time and taking the sum of these values over all departing trains in the plan. Each shunting plan is then given a weight based on this departure delay, such that a plan with much delay is given a lower weight, while the plan is given a higher weight if the delay is closer to zero. This weighting mechanism encourages punctual plans during inference. The final output is the posterior distribution of robust plans for a scenario.

4 Experimental Setup and Results

All experiments were run on a Lenovo Thinkpad E15 gen 3 with the AMD Ryzen™ 5 5500U processor. The experimental setup and configuration files used are stored at the 4TU.ResearchData database[3]. The README file contains all information about the contents of the repository and on how to set up the experiments.

4.1 Importance Sampling on a Large Location

For the first experiments, the 'Kleine Binckhorst' location was used (figure 3), which is a shunting station close to train station The Hague in The Netherlands.

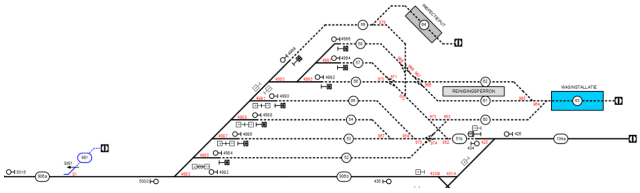

Figure 3: Kleine Binckhorst shunting station by Sporenplan [2]

The scenario has three incoming trains, consisting of four train units in total. Two of these train units are in need of cleaning. The setup of the outgoing trains is the same as that of the incoming trains so recombinations are not included in this example. The initial schedule is composed to be very tight, where the slightest delay of an incoming train can mean a delay in the outgoing trains as well. The arrival times were sampled from a normal distribution centered around these initial arrival times and given a standard deviation of 50 seconds. When running importance sampling for 50 iterations, the weights of the plans in the resulting distribution can be represented in a graph as in figure 4.

To be able to display and analyse the weights, the logarithm is taken and the weights

[3]https://doi.org/10.4121/b974f9f2-672c-44a6-aff8-04806d5f26e6

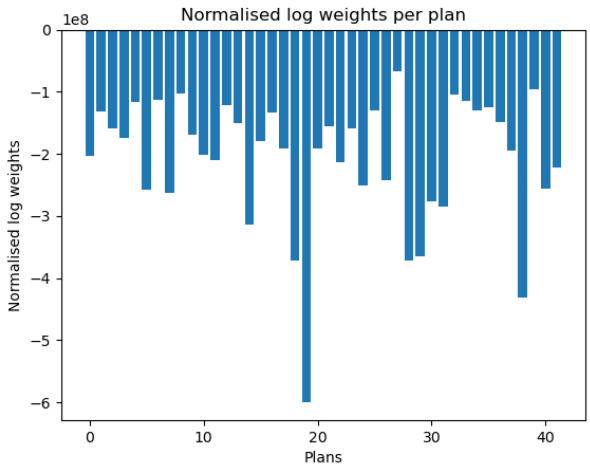

Figure 4: Importance sampling for 50 iterations on the Kleine Binckhorst shunting station involving six trains.

are normalised. A log weight close to zero indicates a plan that has succeeded better in reducing outgoing delay and is therefore more robust. In contrast, increasingly negative values correspond to less robust plans. An interesting remark about the graph is that only 42 plans are displayed. This is because some plans were deemed 'not feasible' by the solver. This highlights one limitation of the solver used, it appears to be unable to generate feasible shunting plans when sufficient train units are not yet present at the scheduled departure time, even in cases where these train units will still arrive later.

4.2 A Simple Shunting Example

The next example will go more in depth into a scenario and show that the robust plans can be an improvement over the deterministic plans generated by the solver. The location and scenario considered are shown in figure 5. In this scenario there are no recombinations or services required, the only task is to route the trains to the departure track in the correct order. Train A arrives first at the shunting yard and is also scheduled to depart first. The trains may use *track_1* or *track_2* to park and wait before they are scheduled to depart, and both tracks are of sufficient length to fit both train A and B at the same time.

Figure 5: Simple routing example where train unit A arrives before B, and departs before B.

Depending on the setup of a planner, it might send both trains over *track_1*. This could work in the initial scenario, but if the trains are scheduled tightly and A is delayed, then B could end up arriving early and blocking train A to reach the destination first. A more robust plan would use both tracks so on no occasion a train will be blocked (figure 6).

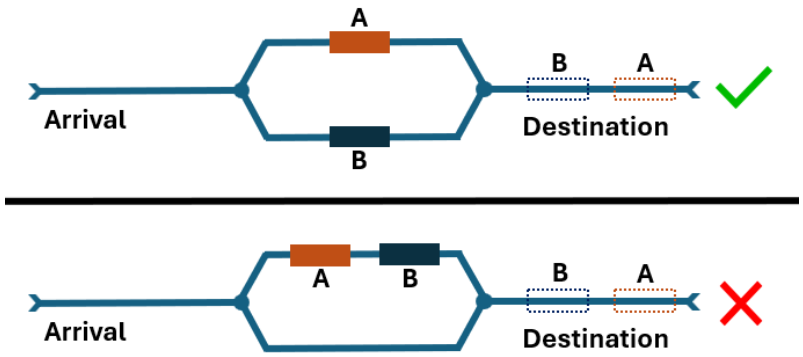

Figure 6: Top: Robust plan, both tracks are used. Bottom: Not a robust plan, if train A is delayed it is stuck behind train B.

In an experiment, the baseline arrival times of the two trains are set on 2050 seconds and 2100 seconds, making it possible to solve the scenario by routing both trains over a single track. The scenario is modelled by sampling the arrival times from a normal distribution with a standard deviation of 60 seconds. Using this setup there is a high chance that scenarios are generated where the first train arrives only after the second train has arrived. Importance sampling is applied on this scenario for 50 iterations. As a result, 38% of the resulting plans divide the trains over both the tracks and the remaining 62% of the plans use only a single track.

4.3 A Shunting Example Including Service Scheduling

As explained in section 2, the TUSP is a complicated problem that involves scheduling, routing and matching, with uncertainties emerging in all these different places. Another example of an uncertainty is the scheduling of cleaning services. A train unit can be scheduled for cleaning, in which case a cleaning duration is given. One train unit might be more dirty than the other and this next example will consider the case where one of the incoming train units has an uncertain cleaning time. As shown in figure 7, the location consists of five tracks, where *track_1* has the ability to clean train units on a cleaning platform. Trains A and B arrive at *track_4* and *track_5*, and should depart on *track_3* and *track_2* respectively. The arrival and departure tracks are connected by a single cleaning track. Train A arrives first and is in need of cleaning. Whilst train A is being cleaned, train B arrives and needs to wait for train A to be finished and clear the track. Train B is scheduled to depart just after train A should be finished with the cleaning task, so right after *track_1* is cleared, it needs to make use of it. The problem with this tight schedule is that as soon as the cleaning of train A is slightly delayed, the initial plan is not valid any more and train B will have to depart with some delay. This can be shown by simulating the setup

and sampling the scenario from a probabilistic model. Whenever the cleaning task finished before the scheduled time, the plan is unchanged. However, often the cleaning task takes longer than expected in which case train B departs with a delay.

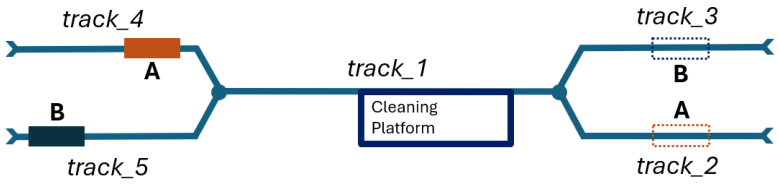

Figure 7: Cleaning example, train A arrives first and needs cleaning. Train A blocks the track for train B which arrives later. If the cleaning of train A is delayed, the departure of train B will be delayed.

5 Responsible Research

5.1 Impact

Transporting humans or goods by trains is still common practice today [5]. When compared to other means of transportation, trains are one of the most efficient and environmentally friendly options [7]. Improving the reliability and efficiency of train transport can support its growth. According to Savelberg and Warffemius, the estimated total costs of train delay in the Netherlands is between 400 and 500 million euros, which is even considered to be an underestimate because some components such as the additional costs in infrastructure management could not be quantified [10]. Needless to say, it would be a great effort if train delays can be reduced and if travelling by train in general can be made more efficient. Railway travel delay substantially affects the emission of trains [16] so minimizing delay would reduce greenhouse gas emissions by itself, but it would also make people and companies more willing to take a train instead of other types of transportation.

5.2 Reproducibility

All the methods, tools, and configuration files used in this study can be found at the 4TU.ResearchData database. This repository includes a README file that explains the structure of the repository, provides links to the configuration files for each experiment done in this research, and describes how to set up the project. The tools can be used on any modern device, ensuring that any result presented in this paper can be reproduced by anyone using the methods and tools listed.

6 Discussion

A remark should be taken into account when discussing the results. This research used a previously developed TUSP solver to study a technique of making train unit shunting plans robust. One of the properties of this solver is that it is not a deterministic solver, like explained in subsection 3.2. In this study, plans are made while incorporating uncertainties

and these are compared to the plan made in the exact, theoretical scenario. Since the solver used is not deterministic, there is already some randomness in the solver itself, which influences the result. This study has tried to reduce the impact of random fluctuations from the solver by increasing the uncertainties introduced, which in the case of random arrival times means a higher standard deviation on the normal distribution used to sample from. Ideally, a deterministic solver should be used when studying the method so any deviation from the deterministic plan can be attributed solely to uncertainties in the input, rather than randomness introduced by the solver.

Figure 4 shows the normalised log weights of the plans when running importance sampling for 50 iterations on a scenario with three trains. This scenario includes routing trains and scheduling cleaning services. The weights have very small values, indicating that the likelihood under the target distribution (the distribution of robust plans) is small. This however is to be expected. The location is relatively complicated, so in spite of the scenario not being too extensive, there are still a large number of different plans possible to be made, resulting in small weights. More interesting are the values relative to one another. One plan stands out as an outlier with a very low value. This indicates that it has low posterior probability under the model, suggesting that it is not a robust plan. When aiming to determine a robust plan it is useful to consider and compare those with higher values.

Two example scenarios are shown in section 4 where the output distribution of the process delivers more robust plans than the original plan. In figure 5 a scenario is considered where an uncertainty in the arrival times can cause a problem where one train is stuck behind a different train, if the two trains are routed to the same track. The output distribution contained several plans where the trains were instead routed on both tracks, making a more flexible plan. Figure 7 showed another example where a train is stuck behind another train. In this case the moves are not so interesting and stay mostly the same throughout all plans. The scheduled timing however does change and therefore the outcomes are most useful for what happens after this scenario. If the train, which is waiting behind the cleaned train, departs with delay it can create large issues on its future timetable. The distribution of robust plans in this case shows that it is desired to leave some margin between scheduled times and is able to uncover where the margin should be: the departure time of the awaiting train should be somewhat later. Both the examples are small and real-world scenarios will often be more complicated, but they do clearly show that the output distribution of robust plans contains plans which are an improvement over the deterministic plan.

This research studied a method for obtaining robust plans for the TUSP, resulting in a distribution of robust solutions. This distribution in itself is not interesting for practical application and it is of interest to investigate how to apply this result in a real-world planning scenario. This research considered the situation at Dutch railways and it is therefore valuable to look at the current state of the planning process. The Dutch railway manager ProRail is working together with the IT company CGI to improve railway scheduling when disruptions occur [1]. CGI developed an algorithm which generates a custom proposal when disruptions occur, consisting of five options. The train dispatcher and the workplace safety leader consequently assess the proposals and decide which one will be used. A similar approach can be applied for using the algorithm worked out in this research. The output distribution can be sampled to obtain a small amount of (distinct) plans. These plans can manually be classified to decide on a final plan to use.

7 Conclusions and Future Work

7.1 Conclusion

This paper aims to create Train Unit Shunting Problem (TUSP) plans which are robust against uncertainties. It builds upon previous work where the method was introduced, and contributes by applying it to a more sophisticated solver capable of handling more complex scenarios, while introducing a new uncertainty not previously explored. The study investigated a method that wraps around an existing TUSP solver and utilises probabilistic programming to make robust plans. The research showed uncertainties which exist in a real-world version of the TUSP and used Gen.jl to implement probabilistic models for both uncertain arrival times and uncertain servicing durations. Next, importance sampling was applied to derive the output distribution of robust plans. The paper proposed a method for applying the output of the process in a real world scenario and two experimental scenarios were demonstrated where it is beneficial to use one of the robust plans over the plan that was initially made by the solver. These scenarios show the potential of integrating probabilistic inference into the planning process in improving railway efficiency and reducing delays.

7.2 Future Work

This research only tried importance sampling as inference method. Gardos Reid suggested to use Markov Chain Monte Carlo (MCMC) methods, where previous shunting plans are reused and updated over time. In his proposed approach, the scenario is updated each iteration by resampling the uncertain variables. Next, the previously created shunt plan is used until the plan is no longer valid, at which point the remainder of the plan is created by rerunning the planner at the current situation. This is a promising technique, as it iteratively alters plans to arrival time deviation, reflecting the real-world process of replanning. This paper did not use MCMC, partly because the solver used did not support reusing old plans out of the box.

The modelling of the problems can be expanded by studying the effect of using different, more realistic distributions to represent delays. Furthermore, the method can be tested on different kinds of uncertainties. The travel speed of a train can differ, or the time it takes for two train units to split or combine can vary. In addition, more impactful uncertainties can be studied, for example a malfunctioning train or switch, resulting in track closure.

This study tried to use an evaluator to assess whether 'robust plans' from the output distribution remain valid on the original scenario. While this attempt was unsuccessful, it would be worthwhile to use a flexible evaluator which is able to evaluate only the sequence of moves, rather than exact timings. This would allow for automatic validation of robust plans against the original scenario, enabling the ability to plan on more complicated scenarios involving more trains, recombinations and service scheduling. Additionally, it would improve quality assessment of the robust plans. A plan can be simulated on several different uncertain scenarios. The plan can subsequently be weighted on the amount of uncertain scenarios it is able to solve, in addition to the amount of output delay they cause.

References

[1] Cgi biedt prorail slimme en snelle hulp bij spoorwegonttrekkingen. https://www.cgi.com/nl/nl/case-study/CGI-biedt-ProRail-slimme-snelle-hulp-spoorwegonttrekkingen. Accessed: 12-06-2025.

[2] Den haag c - binckhorst. https://www.sporenplan.nl/html_nl/sporenplan/ns/ns_nummer/gvc-bkh.html. Accessed: 10-06-2025.

[3] Disruptions on the rails. https://www.ns.nl/en/about-ns/disruptions-on-the-rails.html. Accessed: 20-06-2025.

[4] Gen; an open-source stack for generative modeling and probabilistic inference. https://www.gen.dev/. Accessed: 09-06-2025.

[5] Hoeveel wordt er met het openbaar vervoer gereisd? https://www.cbs.nl/nl-nl/visualisaties/verkeer-en-vervoer/personen/openbaar-vervoer. Accessed: 10-06-2025.

[6] Statistics. https://www.rijdendetreinen.nl/en/statistics. Accessed: 20-06-2025.

[7] Motorised transport: train, plane, road or boat - which is greenest? https://www.eea.europa.eu/highlights/motorised-transport-train-plane-road, 2021. Accessed: 02-06-2025.

[8] Marco F. Cusumano-Towner et al. Gen: a general-purpose probabilistic programming system with programmable inference. 2019. https://dl-acm-org.tudelft.idm.oclc.org/doi/10.1145/3314221.3314642.

[9] Richard Freling et al. Shunting of passenger train units in a railway station. 2005. https://pubsonline.informs.org/doi/abs/10.1287/trsc.1030.0076.

[10] Pim Warffemius Fons Savelberg. Estimation of the social costs of train delays in the netherlands. 2017. https://www.researchgate.net/publication/320881926_ESTIMATION_OF_THE_SOCIAL_COSTS_OF_TRAIN_DELAYS_IN_THE_NETHERLANDS.

[11] Charles J Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992. https://www-jstor-org.tudelft.idm.oclc.org/stable/pdf/2246094.pdf.

[12] Noah D Goodman, Joshua B. Tenenbaum, and The ProbMods Contributors. Probabilistic Models of Cognition. http://probmods.org/, 2016. Accessed: 2025-6-16.

[13] Andrew D Gordon, Thomas A Henzinger, Aditya V Nori, and Sriram K Rajamani. Probabilistic programming. In *Future of software engineering proceedings*, pages 167–181. 2014. https://dl.acm.org/doi/abs/10.1145/2593882.2593900.

[14] E Kleine. Generating robust solutions for the train unit shunting problem under uncertainty: A local search based approach, 2019. https://pure.tue.nl/ws/portalfiles/portal/140069333/Master_Thesis_Esmee_Kleine.pdf.

[15] Amanda Krudde. Detecting patterns in train position data of trains in shunting yards. 2024. https://repository.tudelft.nl/file/File_5cfc78ab-8508-4f85-b69b-613c5670b2f8?preview=1.

[16] H. Christopher Frey M. Grafer. Highway vehicle emissions avoided by diesel passenger rail service based on real-world data. 2016. https://doi-org.tudelft.idm.oclc.org/10.1007/s40864-016-0044-y.

[17] Vikash K. Mansinghka, Ulrich Schaechtle, Shivam Handa, Alexey Radul, Yutian Chen, and Martin Rinard. Probabilistic programming with programmable inference. *SIGPLAN Not.*, 53(4):603â616, June 2018. https://doi-org.tudelft.idm.oclc.org/10.1145/3296979.3192409.

[18] R.J. Gardos Reid. Inferring robust plans with a rail network simulator. 2023. http://resolver.tudelft.nl/uuid:209608e8-9fa7-4e03-aad8-e973ad22cde9.

[19] Surya T. Tokdar and Robert E. Kass. Importance sampling: a review. *WIREs Computational Statistics*, 2(1):54–60, 2010. https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.56.

[20] Jan-Willem Van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint*, 2018. https://arxiv.org/abs/1809.10756.

[21] R.W. van den Broek. Train shunting and service scheduling: an integrated local search approach. 2016. https://studenttheses.uu.nl/handle/20.500.12932/24118.

[22] R.W. van den Broek. A local search algorithm for train unit shunting with service scheduling. 2021. https://pubsonline.informs.org/doi/10.1287/trsc.2021.1090.

[23] R.W. van den Broek. Towards a robust planning of train shunting and servicing. 2022. https://dspace.library.uu.nl/handle/1874/420470.

[24] Jacobus G.M. Van Der Linden, Jesse Mulderij, Bob Huisman, Joris W. Den Ouden, Marjan van den Akker, Han Hoogeveen, and Mathijs de Weerdt. Tors: a train unit shunting and servicing simulator. In *AAMAS '21*, page 1785â1787. International Foundation for Autonomous Agents and Multiagent Systems, 2021. https://research-portal.uu.nl/en/publications/tors-a-train-unit-shunting-and-servicing-simulator-2.

A Use of Large Language Models

Large Language Models (LLM's) have been used to assist the writer during the research. To assist in the writing process, the models ChatGPT-4o, ChatGPT-4o mini and ChatGPT-4.1 mini of OpenAI have been used to improve readability and reduce spelling mistakes. More information about the models can be found on the OpenAI website[4]. An example prompt of the usage of these LLM's is as follows:

**sentence/paragraph of text*
Please highlight any spelling mistakes in this sentence/paragraph of a scientific paper and suggest improved phrasing if necessary.

[4]https://platform.openai.com/docs/models

Changes suggested by the LLM's were verified and if considered an improvement, implemented within the original text.

During modelling and programming, the code completion function of GitHub Copilot has been used to speed up the coding. An example of this is added below in figure 8. The slightly dimmer text after *scheduled_ departure_ times = get*, is a suggestion by Copilot.

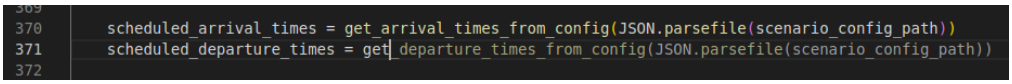

Figure 8: GitHub Copilot Code Completion example