Towards a Robust Planning of Train Shunting and Servicing

Roel van den Broek

Roel van den Broek

Towards a Robust Planning of Train Shunting and Servicing

ISBN: 978-90-393-7483-2

Towards a Robust Planning of Train Shunting and Servicing

Op weg naar een robuuste planning voor het rangeren en onderhouden van treinen

(met een samenvatting in het Nederlands)

Proefschrift

ter verkrijging van de graad van doctor aan de Universiteit Utrecht op gezag van de rector magnificus, prof.dr. H.R.B.M. Kummeling, ingevolge het besluit van het college voor promoties in het openbaar te verdedigen op woensdag 1 juni 2022 des ochtends te 10.15 uur

door

Roel Wemarus van den Broek

geboren op 2 mei 1991 te Tiel

Promotor:	Prof. dr. H.L. Bodlaender
Copromotoren:	Dr. J.A. Hoogeveen
	Dr. ir. J.M. van den Akker

Dit proefschrift is (mede) mogelijk gemaakt met financiële steun
van N.V. Nederlandse Spoorwegen

Acknowledgements

Many people and events contributed to the research presented in this thesis. However, it was set in motion when Han Hoogeveen, assistant professor at Utrecht University, and Bob Huisman, manager of a research department at the Dutch railway company NS, met at a conference. They discussed some of the challenges faced by NS in the planning of railway yards, and Han was confident that solving this problem would be a great topic for a master's thesis. He suggested the train shunting problem to me, which led to a successful master's project, followed by a PhD research position, and ultimately to the thesis you are currently reading.

I would like to express my gratitude to everyone who supported me in the past years. First, I would like to thank my daily supervisors Han Hoogeveen and Marjan van den Akker for their invaluable advice and encouragements, and for inviting me to their home for dinner and board games. Han was always willing to make me a cup of coffee and share his insights into optimization and local search with me. I enjoyed the great discussions that Marjan and I had on robustness in scheduling, which led to a deeper understanding of the subject and many new and interesting research ideas.

Furthermore, I want to thank my promotor Hans Bodlaender for his feedback during the writing of my thesis, and for our conversations during coffee breaks on graph problems, which were a pleasant change from the practical problems within my own research topic.

I would like to thank Rob Bisseling, Dennis Huisman, Sigrid Knust, Marcel van Kooten Niekerk, and Roel Leus for joining the reading committee and offering their comments on this manuscript.

During my time at Utrecht University I had the opportunity to interact with amazing people. I would like to thank all staff members and fellow PhD students for the great times we shared during courses, meetings, lunch breaks and conferences. I want to thank Johan van Rooij for all the inspiring discussions on real-life optimization problems, and Paul Bouman for the fun we had at conferences close to home and on the other side of the world.

This thesis could not have been written without the support of NS. I would like to express my gratitude to NS for funding my PhD-research. Furthermore, I want to thank to all my current and former colleagues at the company for facilitating my research.

In particular, I would like to thank Bob Huisman for creating the conditions to continue my research in a PhD program after my master's thesis. We had many valuable conversations on topics such as the shunting problem, new challenges in the railway sector, and the similarities and contradictions of the scientific and corporate world. Under his supervision I had the freedom to both pursue my own research topics and contribute to the practical applications of my research.

Additionally, I would like to mention Joris den Ouden, Demian de Ruijter, Chris Pustjens, Pim van den Bogaerdt, Wan-Jui Lee and Wouter Fleuren, with whom I had the pleasure of working for most of my PhD research at NS. I am grateful that they embraced me as part of the team, and I would like to thank them for providing me with an inspiring and fun place to work.

Finally, I would like to thank my family, especially my parents Jeanet and Roelof. The last miles of a journey are always the longest, but I could always count on their unconditional support.

Contents

Chapter 1

Introduction

As a result of economical and social development the global demand for transport has grown rapidly over the past few decades, and this trend is likely to continue in the near future for many modes of transportation. Furthermore, due to increasing environmental concerns, the reliance on low-pollution alternatives such as rail will increase for both freight and passenger transport.

One of the major challenges posed by this development is that the growth rate of demand exceeds the rate of expansion of infrastructure, as constructing infrastructure is both enormously expensive and difficult to realize in urbanized areas. Hence, governments and transport companies make an ever-increasing effort to improve the utilization of existing infrastructure. The passenger railway sector, which is the topic of this thesis, attempts to cope with the growing number of passengers by increasing the number of trains moving over the existing railway network.

This focus on better utilization of infrastructure leads to many challenging optimization problems that can no longer be solved (efficiently) by hand. Passenger railway operators have to create and operate denser timetables, and plan the maintenance, movement and storage of more rolling stock.

Significant improvements have been made to the timetabling by the Operations Research community in the last few decades, enabling railway operators nowadays to construct efficient timetables with the help of automated decision support systems. In contrast to these advancements at the railway network level, the fine-grained logistical problem of planning train movements and maintenance at the major stations is still solved manually.

In this thesis, we aim to develop methods that support passenger railway operators in the planning process at major stations and their nearby railway yards. Especially the planning at so-called service yards, where trains are parked, cleaned and maintained during the time that they do not need to drive, is very challenging. It consists of storing surplus trains at the yards, scheduling train maintenance, planning the necessary train movements, and assigning personnel to all tasks. The methods have to be able to construct feasible solutions for real-world planning

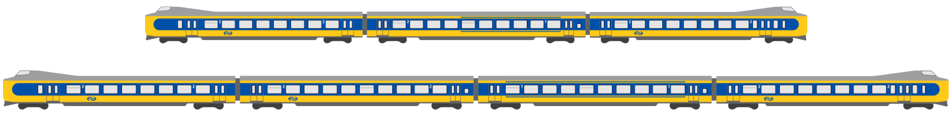


Figure 1.1: Examples of electrical multiple unit trains, both of the type ICM. The sub-types ICM-3 and ICM-4 indicate the number of carriages in the train unit.

problems, and the solutions should be resilient to the day-to-day disturbances that occur on the railway network.

In the rest of this chapter we start by providing a high-level overview of basic concepts and the planning problems that typically arise in the passenger railway sector. We continue with a more detailed description of the train shunting and service scheduling problem in Section 1.2, which is the main focus of this thesis. Literature related to this topic is discussed in Section 1.3. We conclude this chapter with an overview of our contribution to the train shunting problem in Section 1.4 and the outline of the thesis in Section 1.5.

1.1 Passenger Railway Transportation

The goal of passenger railway operators is to efficiently transport passengers over the railway network from their origin to their destination. The railway network consists of train stations, which are the railway hubs of the network, and railway tracks that form the edges of the network by connecting the stations. Passengers are transported over the network by the rolling stock of the passenger railway operators.

Modern rolling stock typically consists of *electrical multiple unit (EMU)* trains, which are self-propelled, permanently coupled carriages with driver cabins on both ends. One of the main advantages of electrical multiple units is that they can move bidirectionally without the need for a dedicated locomotive. We refer to a single electrical multiple unit as a *train unit*. The train units are classified by their *train type*. Further classification into *train sub-types* can be done based on the number of carriages in the train unit, as can be seen in Figure 1.1. Train units of the same train type can be coupled to transport more passengers. A *train* is a group of one or more train units that are coupled. One *train driver* needs to be located in the front unit to operate a train.

Planning problems arise for railway operators both network-wide and inside the individual railway hubs. In the literature the network planning process is usually split into sequential phases:

1. *line planning*: determining the *lines*, i.e., paths over the railway network, that the railway operator will service as well as the servicing frequencies of those lines;

2. *timetabling*: deciding the arrival and departure time of all stops on the lines

based on the servicing frequencies;

3. *rolling stock scheduling*: assigning (compositions of) train types available in the pool of rolling stock to the trips in the timetable.

4. *crew scheduling*: assigning personnel to the trips in the timetable.

In order to solve the high-level network planning problems the logistical challenges within the railway hubs are usually simplified to flow balancing constraints on the train sub-types and storage capacity constraints. The storage capacity provided by some of the hubs is necessary, since the number of trains active on the railway network fluctuates over the day. The number of passengers on a railway network will typically reach its peak in the morning and evening. During these rush hours, most of the rolling stock on the network is used to accommodate the flux of commuters. Outside the peak hours fewer trains are needed to serve all passengers. The resulting surplus of rolling stock is parked off the main railway network at the major stations. The problem of planning the train movements at the railway hubs resulting from the network plan is solved for each of the hubs individually.

1.2 Train Shunting and Service Scheduling

The hubs in the railway network are centered around the major stations and often contain one or more *railway yards*, or *shunting yards*. These railway yards are a collection of tracks, connected by switches, where the rolling stock of passenger railway operators can be stored. In the railway sector, parking trains at the yards is known as *stabling*, and *shunting* refers to moving trains that are not actively in service on the main railway network. The tracks on the shunting yard are classified into *free tracks*, which are accessible from both sides, and *LIFO tracks* that can be accessed from only one direction.

Since the railway yards are located in urban areas, their size, and thus their storage capacity, is very limited. Dense shunting yard layouts are used to exploit the available space efficiently. With storage capacity utilization of up to 90% on some of the yards, this results in highly constrained train movements on the infrastructure.

The process of operating a shunting yard is called *shunting* and has the following three components: matching, parking, and routing. In the matching part we have to assign arriving train units to (positions in) departing trains such that the required *train composition* — an ordered sequence of train sub-types — is satisfied. Train units of the same sub-type are interchangeable in the matching, with the exception of units with scheduled large maintenance. The problem of finding a feasible shunting plan for the combined parking, routing and matching problem, in which every train departs on time from the shunting yard, is commonly known as the *Train Unit Shunting Problem (TUSP)*.

While solving the shunting problem is in itself already a considerable challenge (Lentink et al. (2006)), it is even more difficult for shunting yards that provide additional services. To achieve high passenger satisfaction, regular maintenance and cleaning of trains is crucial. However, due to the dense timetable and the high utilization of both railway lines and rolling stock, passenger railway operators cannot afford to take trains out of service frequently. Therefore smaller service activities, such as cleaning the interior, washing, and maintenance inspections are carried out during off-peak breaks of the trains. This is done at specialized shunting yards, called *service sites.* The service activities are constrained by the availability of resources such as maintenance crews or cleaning installations, and have to be completed before the train departs from the service site. Moreover, they have to take place at specific locations at the service site, which requires additional shunting moves. Shunting plans for service sites have to include a feasible service activity schedule, detailing for each service task when, and by which resource it will be processed. We will refer to the resulting planning problem as the *Train Unit Shunting Problem with Service Scheduling (TUSPwSS).*

Furthermore, the activities in the shunting plan have to be performed by skilled staff members. Personnel is a scarce resource, and hence their availability has a large impact on the feasibility of a shunting plan. Due to the sheer size of a shunting yard, the number of tasks that an employee can perform is severely limited by the walking distances between the locations of consecutive tasks.

The objective in the planning problem is to find a feasible solution, which is a shunting and service schedule that can be executed as planned without violating any physical or safety constraints on the railway yard. Constructing a conflict-free shunting and service schedule by hand is a time-consuming task, even for the experienced planners. Recall that many railway operators are increasing their rolling stock, which makes this task even more complicated. Therefore, automated decision support tools need to be developed to help the human planners cope with the complex planning and scheduling problems at the service sites.

From a computational point of view, finding feasible solutions for the TUSP-wSS is an extremely difficult problem. It combines several well-known NP-hard problems. The service task scheduling can be viewed as an *Open Shop Scheduling Problem* with machine flexibility (multiple identical resources), buffer and blocking constraints (shunting), and release dates and deadlines (based on the timetable). To determine whether all trains can be parked, a *Bin Packing Problem* has to be solved. Furthermore, a parked train is allowed to be reallocated to a different track if, for example, it is blocking another train's movement. Hence, the routing in the shunting plan strongly resembles sliding block puzzles such as the *Rush Hour Problem* (see Flake et al. (2002)). Mathematical programming techniques have been thoroughly investigated for the basic shunting problem (see e.g. Lentink et al. (2006)) with varying success, but typically do not generalize well to the resource-constrained scheduling problems that we have here because of the planning of washing and cleaning; furthermore, these cannot deal with the addition of relocating parked trains.

As challenging as these individual problems are, the algorithmic complexity of constructing shunting and service plans arises mainly from the strong dependencies between the components. The interaction between the different elements makes it practically impossible to effectively decompose the problem into multiple smaller, largely independent problems.

Once the hurdle of constructing a feasible solution is crossed, new challenges arise when implementing shunting plans in an uncertain real-world setting. Disturbances occurring on the railway network result in delayed train arrivals at the shunting yard, and durations of service tasks will inevitably deviate from their norm durations. When these disruptions conflict with the current shunting plan, railway yard operators have to make ad-hoc modifications to the plan. Since disturbances occur frequently, railway yard operators should receive shunting plans that are capable of absorbing most of the small, day-to-day disruptions. To achieve this goal we first have to determine which solution components contribute to the robustness of shunting plans. Then we have to adapt our planning methods with these insights to find solutions that score highly on the robustness metrics.

1.3 Literature Overview

The Train Unit Shunting Problem (TUSP) was first introduced by Freling et al. (2005) and consists of *matching* train units in arriving trains to positions in the departing trains, and *parking* these train units on a track at the shunting yard. The routing of the trains on the shunting yard is not taken into account. The authors use a decomposition approach in which a train unit matching is constructed first. In the matching problem, every train unit in each arriving train is assigned to exactly one position in a departing train, such that the departing trains consist of the correct train types, and the number of times arriving trains have to be split into smaller trains is minimized. The authors formulate the matching problem as a mixed integer linear program and solve it using the standard MIP solver CPLEX. For the parking problem, the authors assume that the arriving trains are split based on the matching on the arrival track and that departing trains are combined on the departure track. Between arrival and departure, the trains are parked on a track at the shunting yard. A column generation approach, with sets of trains that can be parked on the same track as columns, is used to find a feasible parking plan. The authors propose a dynamic programming algorithm to solve the pricing problem. They generated a shunting plan for a typical weekday at the shunting yard in Zwolle, consisting of eighty train units to be parked, in roughly half an hour.

In Lentink et al. (2006), the train unit shunting problem is extended with the subproblem of finding a route over the shunting yard for each train movement. They propose a four-stage approach to construct solutions for this variant of the TUSP. First a matching of train units is determined using the algorithm proposed by Freling et al. (2005). Second they present a graph representation of the physical layout of a shunting yard to estimate the duration of moving a train from its arrival track to some parking track and from there to its departure track. In the third step these estimates are included in the objective of the column generation approach proposed by Freling et al. (2005) to prefer parking assignments with low travel times. Finally, the actual routes are computed by a heuristic using the graph representation and the track occupation resulting from the previous step. The authors have shown that the time needed to generate a feasible shunting plan, including routing, for the shunting yard in Zwolle was around twenty minutes with their approach.

Instead of solving all components of the TUSP sequentially, Kroon et al. (2006) construct solutions for the matching and parking subproblem simultaneously. This greatly increases the complexity of the problem, resulting in a mathematical formulation for the integrated approach that contains a large number of train collision constraints. Testing the model on a realistic case at the shunting yard in Zwolle revealed that there were over 400.000 constraints, which was too much for the CPLEX solver to find a feasible solution in a reasonable amount of time. To reduce the number of train collision constraints, the authors grouped these in clique constraints. This allowed them to find feasible solutions for their test case. Unfortunately, even with the reduction in constraints, the computation time increases rapidly for larger problems, taking several hours to complete.

Several alternative solution methods to solve the TUSP have been proposed by Haahr et al. (2015). They compared constraint programming, column generation and two-staged MIP models with a greedy construction heuristic and a reference MIP formulation on TUSP instances with LIFO tracks. Their results showed that exact techniques are outperformed by the greedy and two-staged heuristics due to excessive memory and computation time requirements.

In all these approaches, the flexibility of a shunting yard is not used to its full extent: parked trains will remain on the same track for the entire duration of their stay at the shunting yard. That is, a train is not allowed to be moved to another location once it has been parked. In Chapter 2 we propose a heuristic that allows trains to be relocated at a different track if that is beneficial to the shunting plan, thus increasing the planning flexibility.

An integrated approach with parking reallocation has been studied by Van den Akker et al. (2008) as well. They propose a greedy heuristic and a dynamic programming algorithm to solve the combined matching and parking problem. The heuristic uses track assignment and matching rules that select the locally best action on arrival and departure such that train units are parked in the correct order for the departing trains. The dynamic programming approach looks at all possible shunting track or matching assignments at each event on the shunting yard, and relies heavily on pruning nodes in the dynamic programming network that are unlikely to lead to the optimal solution as a way to reduce its computation time. In contrast to the model formulated by Kroon et al. (2006), arriving or departing trains are allowed to wait at the platform to avoid conflicts at the shunting yard. Furthermore, the dynamic programming algorithm is also capable of shunting a parked train unit to a different track, resulting in much more flexibility in the shunting plans. This property is difficult to include in the linear programming approach proposed by Kroon et al. (2006), due to the exponential increase in variables and constraints, even when allowing each parking interval to be split only once. The greedy heuristic is quite fast, but it is not capable of finding feasible solutions for complex problems. Even with the pruning rules, the exact algorithm requires more than ten minutes to find a plan for a dozen train units, making it hard to use in practice.

In the work of Lentink (2006) a practical extension to the TUSP is studied. Besides matching, parking and routing, the train units on a service site have to be *cleaned* as well. The cleaning subproblem is a crew scheduling problem, in which each train unit should be cleaned by a crew before it departs from the site. The first three steps are solved using the methods proposed in Lentink et al. (2006). The schedule for the cleaning crews is constructed last. The cleaning problem is modeled as a single machine scheduling problem without preemption, where each cleaning job needs to be finished in a time-window, and the speed of the machine varies over time to reflect the size of the cleaning crew in each shift. A mathematical model based on this formulation, in which the planning horizon is discretized into one-minute-blocks, is solved using CPLEX. In this thesis we extend the cleaning problem to the general problem of scheduling service tasks. Instead of viewing the cleaning as a single machine scheduling problem, we formulate it, together with additional service tasks, as a resource constrained scheduling problem, where the resources are only accessible from a subset of the tracks. See Chapter 2 for a formal description of the service scheduling sub-problem.

An integral approach is used by Jacobsen et al. (2011) to solve a train parking and maintenance problem. Each train has to be maintained at one facility or workshop located on the service site and parked before and after the service task. Using three meta-heuristics, Guided Local Search, Guided Fast Local Search and Simulated Annealing, the authors attempt to construct schedules such that no trains are blocked by other trains, no departure delays occur and the makespan of the service tasks is minimized. Their results show that the local search approaches provide results close to shunting plans constructed by the MIP model, while taking only seconds of computation time compared to the twelve hours needed by the MIP solver. However, the largest instances contain no more than ten trains, with one maintenance task per train, which is not representative of real-world scenarios. The scope of their study is limited to task scheduling and parking. As such, the absence of matching and routing makes it difficult to directly translate their heuristics to the train unit scheduling problem with service scheduling.

1.4 Our Contribution

Whereas the existing techniques from literature described above can often solve sub-problems of the shunting problem reasonably well, constructing feasible solutions for realistic instances of the complete shunting and servicing problem remains difficult with these approaches. Exact models of the problem are challenging to formulate and struggle with large computation times due to the tightly intertwined sub-problems. Furthermore, decomposition heuristics tailored to specific sub-problems are difficult to generalize without a significant drop in performance on the real-world problem instances, in which the railway yards operate close to their maximum capacity.

The main scientific contribution of this thesis is that we present in Chapters 2 and 5 the first algorithm capable of solving the complete Train Unit Shunting Problem with Service Scheduling (TUSPwSS) for real-world instances. It is based on a new partial ordering schedule of the shunting and service plan, as well as a local search algorithm that exploits the partial ordering to find solutions for the TUSPwSS efficiently.

A further contribution of this thesis is the identification of strong predictors of the robustness of shunting plans to small disturbances. As we will discuss in more detail in Chapter 3, the literature is divided over both the definition of schedule robustness and the best metrics to quantify this robustness. We provide theoretical and numerical insights in several existing and new robustness metrics in the general context of scheduling with deadlines under uncertain release dates and task durations, as well as the application to the specific domain of train shunting.

In addition to these scientific contributions, the research in this thesis makes a significant and direct impact on development of automated decision support systems for passenger railway yards in the Netherlands. Firstly, our local search method is already in use by NS to more accurately estimate the logistic capacity of shunting yards in order to support tactical decision making.

Secondly, in close cooperation with NS we have extended the scope of our local search algorithm from single railway yards to major stations with multiple nearby yards. Currently, NS is performing a pilot to evaluate the method on real-life instances of the major station Eindhoven. As preliminary results of both the technical performance and the acceptance by the planners are promising, the local search approach is on track to be implemented on the railway hubs operated by NS in the coming years.

Thirdly, our research on robustness in scheduling can smoothen the implementation of shunting plans at the railway yards by narrowing the gap between "feasible according to the planner" and "feasible in practice". The robustness metrics help planners to evaluate the behavior of shunting plans in an uncertain environment, and the increased robustness of solutions constructed by the local search reduces the necessity of ad-hoc planning by the railway yard operators.

1.5 Thesis Outline

In Chapter 2, which is based on Van den Broek et al. (2021), we provide a formal description of the TUSPwSS without personnel rostering and present an integrated solution method for this planning problem. We describe a partial order schedule representation that captures the full problem, and we present a local search algorithm that utilizes the partial ordering. The proposed solution method is compared to an existing Mixed Integer Linear Program in a computational study on realistic instances provided by the Dutch passenger railway operator NS. We show that our local search algorithm is the first method to solve real-world problem instances of the shunting and scheduling problem. It even outperforms current algorithms when the train unit shunting problem is considered in isolation, i.e. without service tasks. Currently, a real-life pilot performed with planners of NS, and preliminary results show that the planning process can be sped up significantly with the local search approach as decision support system.

To improve the resilience of the shunting and service plans to every-day disturbances, we start in Chapter 3 with an overview of methods to estimate the robustness of solutions to the general problem of scheduling activities subject to temporal and resource constraints as well as deadlines. This problem emerges naturally in numerous application domains such as project management, production planning, and public transport. The schedules often have to be implemented in an uncertain environment, where disturbances cause deviations in the duration, release date or deadline of activities. Since these disruptions are not known in the planning phase, we must have schedules that are robust, i.e., capable of absorbing the disturbances without large deteriorations of the solution quality. Due to the complexity of defining and computing the robustness of a schedule, many surrogate robustness measures have been proposed in literature. In this chapter, we propose new robustness measures and study several properties of both the new and existing measures.

We continue with the subject of robustness in Chapter 4, where we apply the robustness measures to the field of shunting and service planning. Both existing and the newly proposed robustness measures are compared with the results of a stochastic discrete-event simulation study to determine which measures can be applied in practice to obtain good approximations of the true robustness of shunting plans. The robustness measures that showed promising results are then included in the objective function of the local search algorithm introduced in Chapter 2 to investigate their impact on the robustness of the generated solutions. A large number of shunting plans are generated with the local search guided by the different robustness measures for real-world instances of a shunting yard operated by NS, and the robustness of these plans is approximated using the discrete-event simulation to identify which robustness measures guide the local search to highly robust solutions. We show that the addition of a robustness measure based on estimating the completion times significantly improves the robustness of the solutions generated by the local search, outperforming typical minimum slack robustness measures at the cost of computation time. Chapters 3 and 4 are based on the papers Van den Broek et al. (2018) and Van den Broek et al. (2019).

In Chapter 5 we extend the problem in Chapter 2 with personnel rostering. That is, we consider the integration of the staff scheduling into the planning of the railway yards. As the yards often consist of several kilometers of railway track, the main challenge in finding efficient staff schedules arises from the potentially large walking distances between activities. We present two efficient heuristics for staff assignment. These methods are integrated into a local search framework to find feasible solutions to the Train Unit Shunting Problem with service scheduling and staff requirements. To the best of our knowledge, this is the first algorithm to solve the complete version of this problem. On a set of 300 instances of the train unit shunting problem with staff scheduling on a real-world railway yard, the best-performing heuristic integrated into the local search approach solves 97% of the instances within three minutes on average. Furthermore, the integrated approach is also part of the pilot conducted by NS to support their planners.

To further investigate the quality of the solutions produced by the heuristics in Chapter 5, we propose several mathematical formulations based on mixed integer programming and the branch and price procedure in Chapter 6. The bounds obtained by the exact methods for the personnel rostering problem are used to evaluate the heuristic solutions. The two personnel rostering Chapters 5 and 6 are extensions of the paper Van den Broek et al. (2020).

We conclude this thesis in Chapter 7 with an overview of the insights acquired in the preceding chapters, as well as some open problems that can be explored to improve our ability to solve real-world shunting problems.

Chapter 2

Train Unit Shunting with Service Scheduling

2.1 Introduction

In this chapter we consider the Train Unit Shunting Problem with Service Scheduling (TUSPwSS), which is the problem of finding feasible plans for the operation of railway yards with service facilities. The problem originates from NS, the main passenger railway operator in the Netherlands. As the amount of rolling stock is being increased to match the passenger growth and options to expand their railway yards are limited, NS is faced with the increasingly difficult challenge of parking and maintaining their trains on the available shunting yards.

TUSPwSS consists of matching train units arriving on a shunting yard to departing trains, scheduling service tasks such as cleaning and maintenance on the available resources, and parking the trains on the available tracks such that the railway yard can operate conflict-free. The restricted problem of only matching, parking and routing the trains is known as the Train Unit Shunting Problem. The integration of these different planning components, which are in themselves NP-hard problems, leads to a computationally extremely difficult problem.

The goal of this chapter is to develop an algorithm for the construction of conflict-free solutions covering all these aspects of the shunting and service process, thereby supporting the planners of NS in their effort to utilize the existing railway yards more efficiently. To this end we develop a local search algorithm that operates on a partial order representation of solutions to the complete shunting and service problem. Based on the computational study in this chapter we conclude that, as far as we know, this is the first method to solve real-world problem instances of the complete train unit problem with service scheduling.

The remainder of this chapter is structured as follows. We start with a problem description of the Train Unit Shunting Problem with Service Scheduling in Section 2.2. In Section 2.3 we outline our algorithm by formulating a partial ordering schedule of the shunting and service plan that captures the entire planning problem, and we propose local search neighborhoods that operate on this partial ordering. In Section 2.4 the solution method is tested on realistic problem instances based on the service sites operated by NS. We address several challenges that arise when widening the scope of the problem from individual railway yards to major station areas in Section 2.5. Finally, we present our conclusions in Section 2.6.

2.2 Problem Description

The *Train Unit Shunting Problem with Service Scheduling (TUSPwSS)* is an extension of the *Train Unit Shunting Problem (TUSP)* as formulated by Kroon et al. (2006). The main input of the TUSP is a timetable detailing the arrivals and departures of trains and a description of the infrastructural layout of the shunting yard. To include the service scheduling component at the service sites, the input of the TUSPwSS is supplemented with a set of resources as well as the service activities of each train unit that need to be completed before it leaves the service site.

All arrivals and departures of trains on the shunting yard are described by the timetable, and the shunting yard is assumed to be empty before the first arrival and after the last departure. Note that a surplus of rolling stock at the start or end of the planning horizon can be modeled as early arrivals or late departures, respectively. The entries in the timetable consist of the scheduled time of the arrival or departure, the track by which the train will enter or exit the shunting yard, and a specification of the train. The rolling stock of NS consists of bi-directional and self-propelling railway vehicles that move without a dedicated locomotive. Recall from Chapter 1 that train units are classified according to *train type* and *train sub-type*. Train units of the same type can be coupled to form longer combinations; a *train* is a coupled sequence of one or more train units. The sub-type indicates the number of carriages — and thus the length — of the train unit. In TUSPwSS, the level of detail of a train specification depends on whether the entry corresponds to an arrival or a departure. The timetable specifies the exact sequence of physical train units in an arrival, whereas for a departing train, it only indicates the *train composition*, which is a sequence of train sub-types. This provides the flexibility to the planners at the shunting yard to assign a train unit to any position with a matching train sub-type in the departing compositions. The scheduled arrival times in the timetable are assumed to be deterministic, i.e. we assume that all trains will arrive on time.

The service sites operated by NS consist of a set of tracks connected by switches. Tracks can either be dead-end (*LIFO-tracks*) or accessible from both sides (*free tracks*). The *length* of each track indicates the maximum total length of trains that can be parked simultaneously on that track. The duration of train movements is a function of the paths taken by the trains over the shunting yard. This function is part of the input as well. A *service site* also includes a set of resources, such as cleaning equipment or maintenance crews. Each resource can only operate on trains parked on specific tracks.

Each train unit t at the service site has a set of *service activities* that have

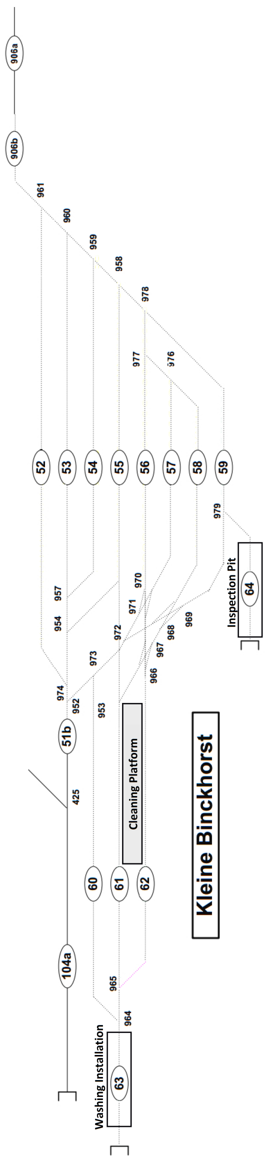


Figure 2.1: The "*Kleine Binckhorst*" shunting yard is operated by NS. The yard contains a washing installation at track 63, a cleaning platform between tracks 61 and 62, and an inspection pit at track 64. Tracks 52 to 59 are used to park trains. The lengths of these parking tracks range from 202 to 480 meters, and each track can contain multiple train units.

to be completed before t leaves the site. Each service activity s for train t has a given processing time $p_{s,t}$ and requires one resource of a specific type for its entire duration. Preemption of activities is not allowed, and each resource can only process a single activity at a time. Furthermore, different service activities of the same train unit or different coupled train units cannot be performed simultaneously. We assume in this study that there are no predetermined precedence relations between the service activities.

The *objective* of the TUSPwSS is to decide whether there exists a feasible shunting and service plan. TUSPwSS consists of the five components below.

1. **Matching**: Arriving train units must be assigned to distinct positions in departing trains such that the train unit type matches the required type in the train composition. All departing trains should leave the shunting yard on time; shunting plans with delayed departures are not feasible.

2. **Combining and Splitting**: As a result of the arrival-departure matching, arriving trains might have to be split and reassembled to form the departure composition. Splitting and combining train units takes time, up to several minutes in practice.

3. **Parking**: During its stay on the shunting yard, whenever a train is not moving, it is parked (or stabled) on some track on the shunting yard. The length of a track should not be exceeded by the total length of trains that are parked simultaneously on it. A train can only depart from the track it is positioned on if it is not blocked by other trains on at least one accessible side of the track. Trains are allowed to relocate during their stay at the shunting yard. Relocating a train requires an additional train movement.

4. **Routing**: For each train movement, the shunting plan should contain a path over the infrastructure. Following the notation of Gallo et al. (2001), a train collision or *crossing* occurs whenever the movement of a train is obstructed by another train. Shunting plans containing crossings are not feasible. The duration of a train movement is determined by its path as well as the driving characteristics of the shunting yard.

5. **Service Scheduling**: All service activities of the train units should be scheduled such that they are completed before their corresponding train unit departs from the service site, and each resource can only process one task at the same time.

In this chapter we assume that sufficient staff members are available at the yard to perform the activities. We relax this assumption in Chapters 5 and 6 by introducing a sixth component to the shunting process, namely the **Personnel Scheduling** sub-problem, and proposing solution methods for this rostering sub-problem.

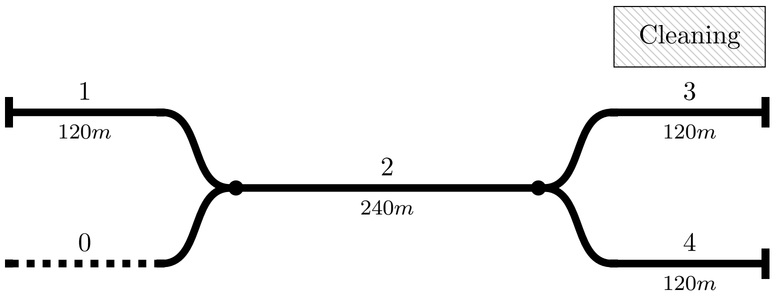


Figure 2.2: An example of a service site. Trains enter and exit the site over track 0 and can only be parked on tracks 1 to 4. The tracks are connected by two switches. The length of the parking tracks is displayed in meters. A cleaning platform allows internal cleaning tasks to be performed on trains positioned on track 3.

Arriving Train	Time
(1, 2)	12:00
(3)	12:45

Departing Train	Time
(ICM-3)	13:00
(ICM-4, ICM-3)	14:00

Table 2.1: The arrivals and departures in the example scenario. The departure trains specify the composition of sub-types instead of the train units, since the assignment is part of the matching problem. The ordering of the train units or sub-types indicates from left to right the order of the train units or sub-types in the train on the service site.

2.2.1 An illustration of the TUSPwSS

To illustrate the complexity of the train unit shunting problem with service scheduling, let us consider a simple scenario of three train units at the service site depicted in Figure 2.2. There are two arriving and two departing trains in this example, which are scheduled according to the timetable in Table 2.1. Note that that the sequences of train units in this table are from left to right. When a train moves to the left side of the shunting yard, the left-most train unit in the sequence is at the head of the train.

The train units are of the ICM type, depicted in Figure 1.1 and described in Table 2.2. Two train units are scheduled for internal cleaning, as can be seen in Table 2.2. In this example, we assume that every train movement takes five minutes. Furthermore, the combining and splitting of trains requires ten minutes.

Recall that to construct a shunting plan we have to decide on

- the assignment of incoming train units to positions in outgoing trains;
- how we are splitting and combining the trains;
- the order of service activities such as cleaning;
- which tracks to move the trains to;

Train Units	Type	Service Tasks
1	ICM-3 (82m)	cleaning (30 minutes)
2	ICM-3 (82m)	cleaning (30 minutes)
3	ICM-4 (107m)	none

Table 2.2: The train units in the example scenario.

- and the order of the train movements.

In our example, it follows from the timetable that the arriving train $(1, 2)$ has to be split into two parts, and that one of the two has to be coupled with train unit 3 to satisfy the requirements of the departing train compositions. Furthermore, with only one cleaning platform, we have to decide on which order we will clean train units 1 and 2.

Let us start the construction of a feasible shunting and service plan by matching incoming to outgoing trains. We assign train unit 2 to the train departing at 13:00; the other two train units will be part of the departing train at 14:00. As train unit 2 is the first to depart, we schedule it to be cleaned first as well, before train unit 1. The scheduled train activities in our shunting plan are listed in chronological order in Table 2.3, and are illustrated in Figure 2.3. In this shunting and service plan, train $(1, 2)$ arrives at track 0 and moves to track 2 to be split. Then train unit 2 heads to the cleaning platform for its service task, and train unit 1 is moved to track 4 to clear track 2 for the arrival of the second train.

After its arrival on track 2, train unit 3 moves to track 1 to avoid blocking the departure of train unit 2. Note that moving from track 0 to track 1 requires two train movements, since the train has to *reverse* its movement direction on track 2. In practice train drivers have to be in the driver's compartment facing the movement direction during a train movement. Therefore, such a *reversal*, which is also known as a *saw movement* or *turnaround*, requires the driver to walk to the compartment at the other end of the train.

When train unit 2 has departed in our shunting plan, train unit 1 goes to the cleaning platform. Both train units 3 and 1 move to track 2 to be combined. Finally, the combination $(3, 1)$ departs from the service site.

This example illustrates the main complexity of the Train Unit Shunting Problem with Service Scheduling. Although the individual shunting sub-problems — matching, combining and splitting, servicing, parking and routing — are seemingly easy to solve, the interaction between these components will make most shunting plans infeasible. Although parking on track 2 is possible, it blocks virtually all routes on the service site. Furthermore, poorly parked trains might require complicated detours, which can easily cause departures to be delayed. The service task schedule is determined entirely by the matching, as there is not enough time to clean both train units of the first arriving train before one of them has to depart. The matching is dependent on the parking and routing as well; switching the order in which train units 1 and 2 depart will result in an infeasible solution due to the small time-window between the first arrival and departure.

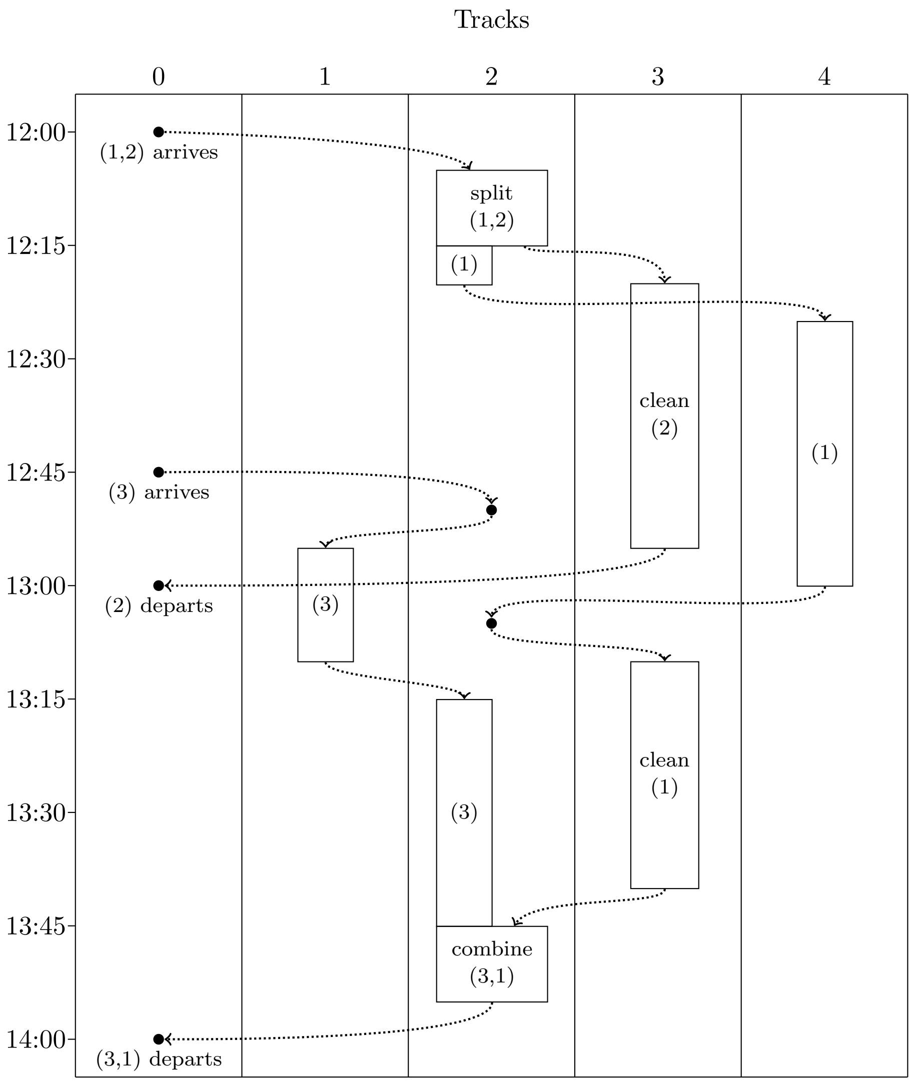


Figure 2.3: An overview of the positions of trains over time in the shunting plan listed in Table 2.3. Dotted lines represent train movements.

Start	End	Train	Activity	Tracks
12:00	12:05	(1,2)	Arrival	$0 \rightarrow 2$
12:05	12:15	(1,2)	Splitting	2
12:15	12:20	(2)	Movement	$2 \rightarrow 3$
12:20	12:50	(2)	Cleaning	3
12:20	12:25	(1)	Movement	$2 \rightarrow 4$
12:45	12:50	(3)	Arrival	$0 \rightarrow 2$
12:50	12:55	(3)	Movement	$2 \rightarrow 1$
12:55	13:00	(2)	Departure	$2 \rightarrow 0$
13:00	13:05	(1)	Movement	$4 \rightarrow 2$
13:05	13:10	(1)	Movement	$2 \rightarrow 3$
13:10	13:40	(1)	Cleaning	3
13:10	13:15	(3)	Movement	$1 \rightarrow 2$
13:40	13:45	(1)	Movement	$3 \rightarrow 2$
13:45	13:55	(3)	Combining	2
13:55	14:00	(3,1)	Departure	$2 \rightarrow 0$

Table 2.3: The train activities in a shunting plan for the example scenario provided in Tables 2.1 and 2.2.

2.3 Local Search Heuristic

To find feasible solutions for the Train Unit Shunting Problem with Service Scheduling, we propose a local search approach that includes the full problem, i.e. it integrates the matching, combining and splitting, parking, service scheduling and train movement components of the planning into a single model. Local search algorithms gradually improve some candidate solution, a shunting service plan in case of the TUSPwSS, by making small changes to it, and have been applied in the field of Operations Research with great success. Methods to create these changes are called *(local search) operators*, and the set of solutions attainable from the current solution by the same operator is known as the *neighborhood.*

Essential to any local search algorithm is a solution representation that properly captures all important aspects of the solution, while simultaneously allowing for easy modification through the local search operators and efficient evaluation of the objective. *This is especially important as well as challenging for the TUSPwSS, because of the complex structure of its solutions and its tightly intertwined subproblems.* We will model each activity in the shunting plan as a node in a precedence graph. We will refer to the resulting directed acyclic graph as the *activity graph*, which is a partial order schedule of the activities. The main challenge is that the graph should be updated efficiently and must remain acyclic after applying an operator.

When solving TUSPwSS, we are facing the decision problem of finding feasible shunting and service plans, where feasibility is difficult to achieve because of the high utilization factor of the yard. To alleviate this difficulty, we transform the decision problem of finding feasible shunting and service plans into an optimization problem by relaxing some of the feasibility constraints and apply local search on the resulting problem. Instead of enforcing that these relaxed constraints are respected in all solutions explored by the local search, we penalize violations of the constraints in the objective function. A shunting and service plan constructed by the local search is then feasible if and only if none of the relaxed problem constraints are violated.

We have based our algorithm on the *Simulated Annealing* framework by Kirkpatrick et al. (1983) and Černỳ (1985), which is a stochastic local search technique that has seen many successful applications to other combinatorial optimization problems. A simulated annealing algorithm randomly selects a neighbor 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. Let b be the selected neighbor of the current solution a, and suppose we are minimizing an objective function f. If $f(b) > f(a)$, then the probability of acceptance P is

$$P = e^{\frac{f(a)-f(b)}{T}}, \tag{2.1}$$

where T is a control parameter that will be decreased during the search to accept less deterioration in solution quality later on in the process. See Section 2.4 for an overview of other parameters of the simulated annealing relevant to the computational experiments.

In the following we will present the objective function and the distinction between hard and soft constraints that we apply to find a feasible solution. Then we explain the representation of the solution by an activity graph and after that we discuss the local search operators. Finally, we describe the construction of an initial solution.

2.3.1 Objective Function and Constraints

Recall that we transform the decision problem of finding feasible shunting and service plans into an optimization problem by relaxing some of the constraints and penalizing violations of these constraints in the objective function. The relaxation of constraints is a trade-off between the size of the solution space and the ease of exploration of the solutions in the local search. Therefore, the decision on which constraints to relax largely defines the structure of the solution space that the local search will explore. In the remainder of this subsection, we start by providing a summary of the problem constraints. Then we motivate the relaxation choices that we make in our proposed method, and we conclude with an overview of the objective function.

We categorize the constraints of the shunting and service problem in four groups, namely

- **matching**: assign incoming train units to outgoing trains, splitting and combining the trains if necessary;

- **sequencing**: find an order for the activities that share the same service resource or movement infrastructure;
- **temporal**: ensure that trains can enter the yard directly upon arrival and depart on time;
- **parking**: park the trains without exceeding the track capacity or blocking train movements.

Constructing an assignment of arriving train units to departing trains that satisfies the **matching** constraints is not a difficult problem in itself. Moreover, any mutation of the matching — regardless of the feasibility of the resulting assignment — will likely have a large impact on the entire shunting and service plan, as it affects the parking, movement and maybe also servicing components of the solution. Therefore, we keep the **matching** constraints strict, guaranteeing that any solution will have a feasible matching.

Imposing both the **sequencing** constraints of the service tasks and the **temporal** constraints on the train departures as hard constraints makes it difficult to find a feasible schedule for the service tasks. This implies that this combination of constraints severely restricts the number of candidate solutions that can be reached efficiently during the search and hence relaxing some of the constraints will be beneficial to the search process. In our approach we maintain the **sequencing** constraints as hard constraints, and relax the **temporal** constraints. That is, we allow the local search to construct shunting plans with delayed trains as intermediate solutions at the cost of a penalty.

Similarly, imposing the combination of **sequencing** and **parking** constraints on the train movements creates a subproblem similar to computationally difficult sliding block problems such as RushHour. Furthermore, the **parking** constraints on the track capacity imply that a Bin Packing problem has to be solved in each iteration, which becomes difficult in instances with a large degree of utilization of the shunting yard.

Violations of the relaxed **temporal** and **parking** constraints are penalized in the objective function. For a shunting and service plan p, define $Delay(p)$ as the number of delayed entering and departing trains, $Crossing(p)$ as the number of crossings (i.e. collisions), and $TrackCapacity(p)$ as the number of occasions in which the combined train length of trains parked on a track τ exceed the capacity l_τ. An arrival delay will occur when an arriving train cannot move immediately from the arrival track to its parking location due to a movement of another train. These characteristics are used to quantify the weighted number of constraint violations, denoted by violations(p), of the shunting and service plan as

$$\begin{aligned} \text{violations}(p) = \quad & w_{delay} \cdot Delay(p) + w_{crossing} \cdot Crossings(p) + \\ & w_{track} \cdot TrackCapacity(p), \end{aligned} \tag{2.2}$$

where each type of violation is multiplied by its corresponding weight $w > 0$, and p is feasible only if violations$(p) = 0$.

Notation	**Description**
$\mathcal{A}$	Set of train activities in a shunting plan
$\mathcal{POS}$	Partial order schedule
$\mathcal{M}$	Set of movement activities, $\mathcal{M} \subseteq \mathcal{A}$
t_a	Train associated with activity $a \in \mathcal{A}$
o_a	Start location of activity $a \in \mathcal{A}$
d_a	Final location of activity $a \in \mathcal{A}$
r_a	Resource required by activity $a \in \mathcal{A}$

Table 2.4: Overview of the notation used to describe the shunting plans.

Although the expression above can be used directly as the objective of the local search, we extend the objective function with several additional terms to guide the local search to more promising regions in the solution space. The cost function minimized in the objective of our approach is

$$\begin{aligned} \text{cost}(p) = \quad & \text{violations}(p) + w_{time} \cdot \text{TotalDelayTime}(p) + \\ & w_{movement} \cdot \text{NumberOfMovements}(p), \end{aligned} \tag{2.3}$$

which penalizes the severity of a delay in addition to the occurrence of violations. Furthermore, shunting plans with fewer movements are both preferred by the planners at NS and easier to improve by the local search. Therefore, we include the number of movements in the objective with weight $w_{movement}$, which is chosen small enough to never prefer a reduction in the number of movements over the resolution of a conflict.

2.3.2 Solution Representation and Evaluation

Our representation of a shunting plan in the local search procedure consists of a set $\mathcal{A}$ of *train activities* and a set $\mathcal{POS}$ of *precedence relations* that defines a *partial order schedule* on the train activities. See Table 2.4 for an overview of the notation used in this section.

The activity set consists of four types of activities: *arrival*, *departure*, *service*, and *movement*. The precedence relations arise from the sequencing constraints. These constraints enforce that activities of the same train or on the same service resource do not overlap. Moreover, they forbid conflicts between two moving trains. Making sure that a solution satisfies these constraints boils down to sequencing activities, i.e. imposing precedence relations. Now we obtain the *activity graph*, which is a directed graph whose nodes are the activities and arcs are the precedence relations.

Each activity $a \in \mathcal{A}$ is associated to a train t_a, which is an ordered list of train units. We will refer to the set of all train movement, arrival and departure activities as the *movement activity set* $\mathcal{M} \subseteq \mathcal{A}$, with for each $a \in \mathcal{M}$ an origin o_a and a destination d_a. Note that an arrival or departure activity represents a train movement from or to the main railway network, respectively. Each service activity

s is associated with a resource r_s; its location is the destination d_a of its predecessor movement. The splitting and combining of trains is modeled implicitly in the data structure by a difference in the train compositions of the trains associated with subsequent activities. The representation of the example shunting plan described in Section 2.2.1 is shown in Figure 2.4.

In the activity graph, the paths taken by the train movements and the start time of the activities are not included explicitly. To keep the routing computation tractable, the local search generates activity graphs with a total ordering on the activities in the movement activity set $\mathcal{M}$, such that never two movements overlap in time. Consequently, we can compute routing and time assignment in two steps, respectively. In the first step we strictly enforce the total ordering of the movements. We relax this restriction in the second step to allow trains to move simultaneously as long as this does not result in conflicts.

In the first step of a solution evaluation, we compute a path for each movement activity separately and determine the number of crossings and track capacity violations. To achieve this, we iterate over the movement activities according to the total order in the activity graph. For each movement activity a, we compute the minimum cost path of train t_a from o_a to destination d_a. Note that, due to the restriction of the movement ordering in the partial order schedule to a total ordering, all movements occur sequentially. Therefore, we can formulate the routing problem of a single train movement as a single-source shortest path problem in a graph representation of the shunting yard similar to the approach taken by Lentink (2006). The cost of a path in this graph is equal to the time it takes for train t_a to move over the path, plus the number of crossings — i.e., collisions with parked trains — that occur along the path times a weight λ, where λ is sufficiently large to ensure that the number of crossings is minimized. To find shortest paths we apply the $A*$ algorithm (Hart et al. (1968)), where we use the path durations in the static case without parked trains as the lower-bound heuristic on the true path cost. After computing the path of the movement, we add train t_a to the destination track d_a and update the track capacity violation count if necessary.

In the second step, we assign start times to all activities in the activity set. Due to our restriction on the partial order schedules described earlier, all train movements would be scheduled sequentially, which could result in many delayed departures in the shunting plan. To decrease unnecessary delays, we relax the precedence relations between pairs of train movements. Then, for each activity $a \in \mathcal{A}$, we compute its start time as the maximum over its release date (if it is an arrival) and the completion time of all its direct predecessors in the activity graph. More specific, when we consider movement activity a, we compute the earliest possible starting time of a such that

- a starts after the completion times of all scheduled activities, i.e. activities that have already been assigned a time-stamp, that have a train unit in common with train t_a;
- the path of a does not intersect with the path of any scheduled movement a' that happens at the same time;

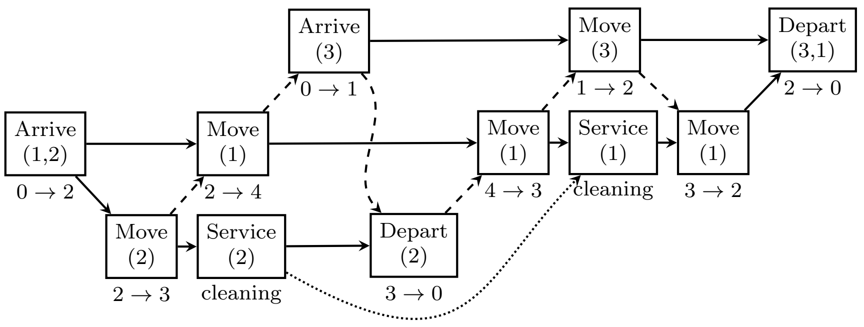


Figure 2.4: The partial order schedule of the shunting plan described in Section 2.2.1. The nodes represent train activities, with the corresponding train between parentheses, and the arcs indicate precedence relations of activities that require the same train unit (solid arcs), service resource (dotted), or movement infrastructure (dashed). The $i \to j$ notation below a node indicates a movement from track i to track j.

- a does not cause additional collisions or track capacity violations.

The last constraint is necessary to prevent a deterioration in solution quality due to the relaxation of the total ordering on the movements.

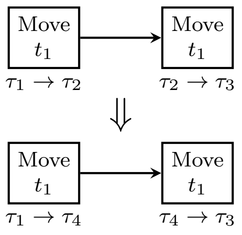
Many shortest path problems have to be solved in each candidate solution evaluated by the local search, as even small, local changes to the shunting plan can affect multiple train movements. In our approach, we only recomputed the paths of movements that might have been affected by the application of an operator in the iteration.

2.3.3 Search Neighborhoods

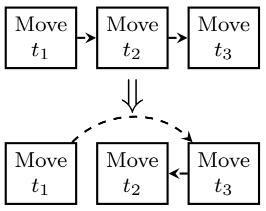
In the local search framework, new candidate solutions are selected from search neighborhoods centered around the current solution. To address the different aspects of the train unit shunting problem with service scheduling, we propose several search neighborhoods that are tailored to the different components of the planning. The corresponding operators either change the location of a train in the plan, or alter the activity graph directly by adding and removing vertices or arcs.

To avoid deadlocks, the application of an operator to the current solution must preserve the acyclicity of the partial order schedule. As a result of the dependencies between the problem components, this means that when we change the shunting and service plan in one dimension, we also need to modify the plan in other dimensions. For example, if we change the service schedule, then the train movements have to be adapted accordingly.

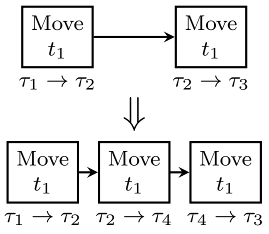
We will now provide an overview of the proposed local search neighborhoods for the parking, routing, service scheduling and matching components. The splitting



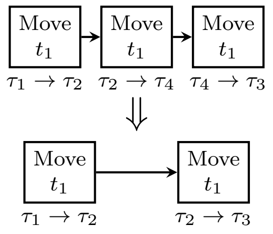
(a) Change the parking location.



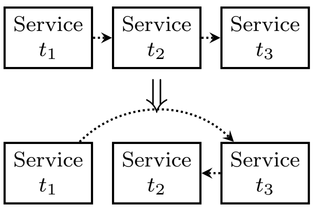
(b) Change the movement ordering.



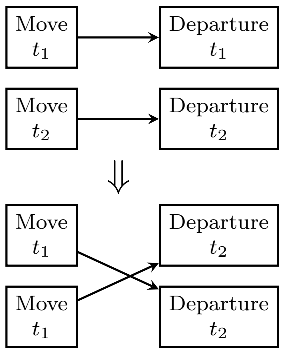
(c) Insert a train movement.



(d) Remove a train movement.



(e) Change the schedule of the service activities by assigning a service activity to a different compatible resource or changing the activity ordering within a resource.



(f) In the matching, swap the assignment of two incoming trains to outgoing train compositions.

Figure 2.5: Overview of neighborhoods in the proposed local search method. t_i denotes train i, $\tau_j \to \tau_k$ indicates a train movement from track τ_j to track τ_k.

and combining activities follow implicitly from the other activities, and therefore have no dedicated local search neighborhoods. For each of the proposed neighborhoods that might contain solutions with cyclic precedence relations, we will show the methods implemented to restrict the neighborhood to acyclic solutions. An overview of the neighborhoods is shown in Figure 2.5.

Parking

Conflicts in the shunting plan such as crossings and track capacity violations can often be solved through changes in the parking location of trains. The **track assignment** neighborhood consists of all shunting plans that can be constructed by changing the track on which a train is parked. To change the location of a train, we select two consecutive movements m_1 and m_2 of the train, and assign both the destination of m_1 and the origin of m_2 to a different track, as shown in Figure 2.5a. If the train is split into several smaller trains after m_1, then the next train movements of all the parts have to be updated. Similarly, if m_2 is preceded by a combine activity, then all predecessor train movements of the different train parts need to be updated as well.

Train Movement

The paths taken by the trains are recomputed whenever the track occupancy changes, and as such, no local search operator is needed for the path-finding component of the routing problem. However, as we maintain a linear ordering of the movements in the partial order schedule, we can attempt to improve a shunting plan by reordering the movements. Suppose that train a is parked on a LIFO-track. If train b arrives on the same track just before train a departs, a crossing will occur. In this case, it is beneficial to let a depart before b arrives. The search neighborhood of rearranging movements is denoted as the **shift movement** neighborhood. The corresponding local search operation, depicted in Figure 2.5b, consists of selecting a movement activity and shifting it earlier or later in the linear ordering imposed on the train movements. To ensure that the resulting shunting plan is valid, only shifts that preserve the acyclic property of the partial ordering are included in the search neighborhood.

In some cases, we want to move a train temporarily to a different track. For example, if train a has to move over track τ while train b is parked there, then one approach to resolve the planning conflict is to move train b to a different track just before the movement of a. In the partial order schedule this operation corresponds to inserting an additional movement activity for train b, visualized in Figure 2.5c. The **insert movement** neighborhood consists of all solutions obtainable by adding a movement activity.

Conversely, it can also be beneficial to remove redundant train movements. Suppose that a train in the shunting plan has a service activity on track τ_1, then moves to track τ_2 for parking, before continuing to track τ_3 for another service activity. If we could skip the parking and move straight from track τ_1 to τ_3 without conflicts, then we have eliminated a movement activity, resulting in more temporal flexibility for other train movements. Solutions in the **remove movement** neighborhood are constructed by removing a train movement from the solution, see Figure 2.5d.

Service Scheduling

The local search operators that adjust the resource assignment and order of the service tasks are based on operators proposed in the literature on similar problems such as the Job Shop Problem (Dell'Amico et al. (1993)), the Open Shop Problem (Liaw (1999)) and their generalized counterparts (Bürgy et al. (2011)). All valid solutions that can be constructed by swapping the order of two consecutive service tasks, that are either assigned to the same resource or involve the same train, are part of the **service order swap** neighborhood, see Figure 2.5e. Furthermore, the **resource assignment** neighborhood contains the solutions obtained by assigning a single service activity to a valid position in the activity schedule of a different suitable resource. Observe that rescheduling service activities often requires adjusting the precedence relations of movements from and to these service activities.

Matching

The **matching swap** operator, shown in Figure 2.5f, changes the matching of incoming trains to outgoing departure compositions. It selects two trains t_1 and t_2 in the shunting plan of identical train composition and swaps their assignment to the departing trains.

2.3.4 Initial Solution

To construct a starting point for the local search, we propose a simple sequential algorithm for the TUSPwSS. We start with the matching subproblem. A perfect matching between the incoming and outgoing train units is constructed such that no arriving unit is matched to a position on a train that departs before all service tasks of the unit can be finished. Note that we can immediately abort the search for a feasible shunting plan if no perfect matching is found, as the existence of such a matching is a necessary condition for plan feasibility.

From the train unit matching we can derive the minimum number of splits and combines that have to be performed to transform the incoming trains into the desired departure compositions. Train units coupled on arrival can only remain together if

1. all units are assigned in the same order to consecutive positions on a departing train,
2. their arrival time plus the sum of the durations of their service tasks is no more than the departure time, and

3. for each service task there is a track adjacent to the required facility that is long enough to harbor all train units at once.

Based on this information, we initialize the partial order schedule:

1. The arrival and departure nodes are added to the solution;
2. For each arrival activity, we add a movement to the graph and connect it to the corresponding arrival node. Similarly, movements are added to each departure node;
3. The movements from arrivals and to departures are connected by arcs to reflect the matching, splitting and combining computed above.

In the next step, we construct a service schedule. The service activities of a train will be scheduled after it has been split, and before it will be combined in the current plan. The service tasks are scheduled by a list-scheduling strategy. Trains are sorted on increasing departure time. Service tasks of the same train are assigned to the resource that becomes available at the earliest time. If a task can be assigned to multiple resources, the resource with the currently smallest total workload is selected. Ties in the train task order are broken randomly. The service activities are then inserted into the partial order schedule and connected with arcs based on the precedence relations in the service schedule.

Next, we add movement activities to and from each service activity to the graph, as trains have to be able to reach the service facilities. The linear order of movement activities is constructed by sorting the movements by earliest starting time, based on the service task schedule.

Finally, the parking locations of the trains are assigned. For every train, we select a random track long enough to store the train for each parking time-window between consecutive movements, without taking the track occupation into account.

2.4 Computational Results

In this section we study the performance of the proposed local search approach on generated test cases as well as a real-world problem instance. These instances are based on two shunting yards that are considered difficult by the planners of NS due to the high degree of utilization of the yards in practice.

Preliminary tests were conducted to obtain good parameters for our local search. In all experiments we have conducted, the local search continued searching until either a feasible solution was found, or a maximum computation time of five minutes was reached. The maximum computation time is based on preferences of NS. The control parameter T of the simulated annealing decreased exponentially in those five minutes, starting at 1 and dropping to 0.01 after 300 seconds. The weights of the objective function used in the experiments are summarized in Table 2.5. Delays are penalized more than the other conflicts, as these were observed to be more difficult to resolve by the local search. Furthermore, the weight of train movements is small with respect to the weights of the conflicts to ensure that conflict resolution is prioritized over the reduction of the number of train movements. Table 2.9 shows some results of experiments with different parameter values. The local search procedure randomly selects a candidate solution in a neighborhood and either accepts or rejects it based on its acceptance criterion. The computations were performed on a computer with an Intel Xeon E5 3.0 GHz processor.

We will compare our simulated annealing approach with a mixed integer programming heuristic developed by NS for the TUSP, i.e., only the matching, parking and routing sub-problems. This tool, called OPG, first computes the routing duration of shunting from one track to another, similar to the approach taken by Lentink et al. (2006). Secondly, the matching and parking sub-problems are solved simultaneously, using the cost estimates of the routes to find track assignments that simplify the subsequent routing problem. To find a matching and a parking assignment, a problem formulation based on the mathematical model introduced by Kroon et al. (2006) is solved in CPLEX. In the final step of OPG a MIP model is solved to assign starting times to all train movements. The mathematical models in OPG lack the flexibility of the local search algorithm to schedule service tasks or insert additional parking activities. That is, OPG keeps a train at the same location during the entire interval between the arrival and departure of the train. As with the solution method proposed in this chapter, we limit the maximum computation time of OPG to five minutes. Since OPG is not capable of solving the service scheduling sub-problem, we compare the two approaches on instances without service tasks. OPG finished its computations within the maximum time for almost all tested instances.

Weight	w_{delay}	$w_{crossing}$	w_{track}	w_{time}	$w_{movement}$
Value	2	1	1	0.00025	0.01

Table 2.5: The weights of the components of the objective function in Equation (2.3) used in our experiments.

2.4.1 Real-world Scenario

We have tested our solution method on one of the real-world instances currently planned manually at NS. The test scenario is a normal week day of twenty-four hours at the "*Kleine Binckhorst*", shown earlier in Figure 2.1. The Kleine Binckhorst is a medium-sized service site situated near The Hague Central Station, and consists mostly of tracks accessible from both sides. Tracks 906a and 104a connect the Kleine Binckhorst to the main railway network; parking, reversing, splitting and combining on these tracks are not allowed due to safety regulations. Tracks 52 to 63, with lengths in the range of 192 to 473 meters, are available for parking. There are two dedicated service facilities: a washing machine on track 63, and a platform for internal cleaning between tracks 61 and 62. Only a single train can be cleaned externally at the washing machine at the same time. There are two crews

Train type	Reversal base duration	Reversal duration per carriage
SLT	2	$\frac{1}{3}$
VIRM	4	$\frac{1}{2}$
DDZ	4	$\frac{1}{2}$

Table 2.6: The reversal duration of the train types in minutes. The duration consists of a base time required to transfer control and additional walking time per carriage.

Sub-type	length	Cleaning	Washing	Maintenance check
SLT-4	70	15	23	23
SLT-6	101	20	24	27
VIRM-4	109	37	24	11
VIRM-6	162	56	26	14
DDZ-6	154	56	26	18

Table 2.7: The train length in meters and the service task duration in minutes for each train sub-type.

at the cleaning platform, allowing a train to be cleaned at each track adjacent to the platform. The maintenance checks that are carried out by service crews at Kleine Binckhorst can take place on any track that is not part of some facility. The reversal duration of trains and the average service task duration are listed in Tables 2.6 and 2.7. The duration of a movement is computed using the following Equation (2.4)

$$d_{\text{driving}} = N_{\text{tracks}} + \frac{1}{2}N_{\text{switches}}, \tag{2.4}$$

where N_{tracks} and N_{switches} are the number of tracks and the number of switches on the path of the movement, respectively.

The instance that we considered consists of 32 train units, arriving and departing in 23 and 21 trains, respectively. Due to the timetable, the maximum number of train units simultaneously present on the service site is 25; these train units occupy 77 percent of the total track length available for parking. There are 59 service tasks that must be completed: 27 internal cleanings, 25 maintenance checks and 7 train washes. Constructing a shunting and service plan for this instance by hand usually takes more than an hour, even for an experienced planner.

We have used the simulated annealing approach described above to search for a feasible plan for the test case, which was found after four minutes of computation time. In this solution, the 23 arriving trains are split into 27 trains. The shunting plan contains 88 shunting movements, of which 32 contain a reversal. The large number of reversals results in an average movement duration of 10 minutes. This means that almost 15 hours of train movements are needed in this 24-hour shunting plan. In 14 cases a parked train is shunted to a different track to make room for another train.

Sub-type	Arrival	Cleaning	Washing	Maintenance check
SLT-4	0.28	1.00	0.16	1.0
SLT-6	0.17	1.00	0.16	1.0
VIRM-4	0.41	1.00	0.16	0.58
VIRM-6	0.10	1.00	0.16	0.58
DDZ-6	0.04	1.00	0.16	0.58

Table 2.8: The Arrival column shows the distribution of the train sub-types over the arriving trains. The probability that a task has to be performed on a certain train unit is shown in the last three columns.

The shunting and service plans constructed by the algorithm differ significantly from the manually created plan. The solution of the planners clearly shows a high-level strategy of first doing the maintenance checks on tracks 56 to 59, followed by internal cleaning and washing, before parking the trains on tracks 52 to 55 until their departure. In contrast, the plan produced by the simulated annealing utilizes the tracks and resources of the service more evenly.

2.4.2 Generated Instances

To evaluate the performance of the proposed solution method more thoroughly, we generated problem instances for two service sites operated by NS. These instances vary in the number of train units that arrive, but resemble real-world scenarios at the service sites in all other aspects. For example, the train compositions and timetable of arrivals and departures, as well as the required service activities of train units are drawn from distributions fitted to historical data of the two service sites. The planning interval of the instances is limited to the night shift, from 6 p.m. to 8 a.m., as most activities at a service site take place during the night. Furthermore, all nightly arrivals occur before the first departure in the morning, which means that the maximum number of train units simultaneously present at the service site in a problem instance is precisely the total number of arriving train units. The train type and service task distributions used to generate the instances are shown in Table 2.8. The train lengths and the service durations are as in Table 2.7. The maximum length of composite trains in our test cases is three train units, and approximately half the arriving and departing trains are composed of two or more train units.

One of the two tested service sites is the Kleine Binckhorst. For every $k \in \{4, 6, \ldots, 32\}$, we generated 50 instances for the Kleine Binckhorst with k train units. These instances are not necessarily all feasible, particularly the instances with many train units are likely to be impossible to solve. When all train units have to be cleaned internally, the planners at NS estimate the capacity of Kleine Binckhorst at roughly twenty train units during the night shift.

Since we want to compare to OPG, which does not contain parking relocation, we investigate the impact of the parking relocation neighborhood on the perfor-

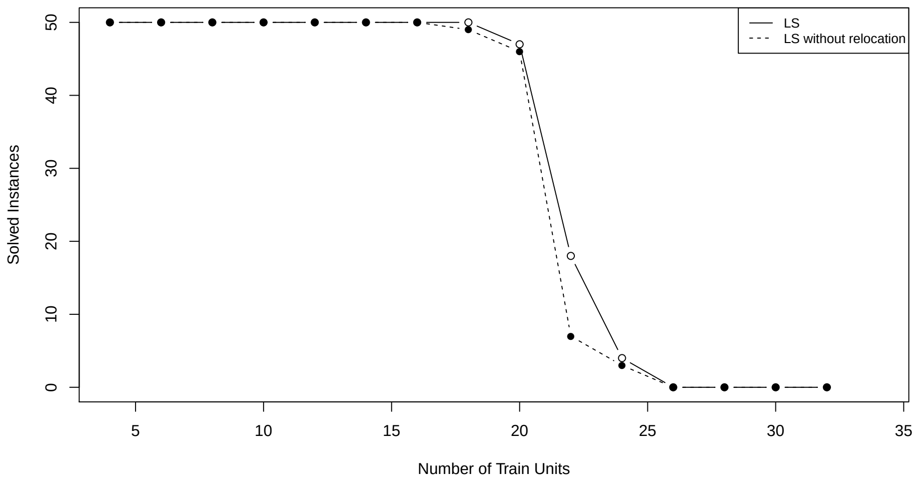


Figure 2.6: The number of feasible shunting plans found for each set of fifty night-shift instances of the Kleine Binckhorst. The results of both the simulated annealing with the relocation operator (*LS*) and without (*LS without relocation*) are shown.

mance of the local search approach. We ran the simulated annealing algorithm with and without the parking relocation neighborhood on all instances. Both variants of the local search method were run with the settings described in Table 2.5.

The results of the experiments are shown in Figures 2.6 and 2.7. The local search method is able to plan up to 18 train units reliably, and fails to solve instances with more than 22 train units. Removing the parking relocation operator from the local search does not result in significant deterioration of the performance. Both simulated annealing approaches show a gradual rise in computation time, requiring less than half the allotted time of five minutes to solve instances with at most 20 train units.

The similarity of the performances of the two local search variants is likely caused by the number of service activities, as these activities force the trains to move to or from service facilities, allowing the local search to solve conflicts in the parking component by changing the service schedule. To test this hypothesis, we generated similar instances without service tasks. Since only the matching, combining and splitting, parking and routing problem components remain, these instances are essentially TUSP instances. We performed the same experiments with the two simulated annealing variants as above; the results can be found in Figure 2.8.

In a pure TUSP instance the simulated annealing variant without the parking relocation neighborhood performs significantly worse. Without the additional

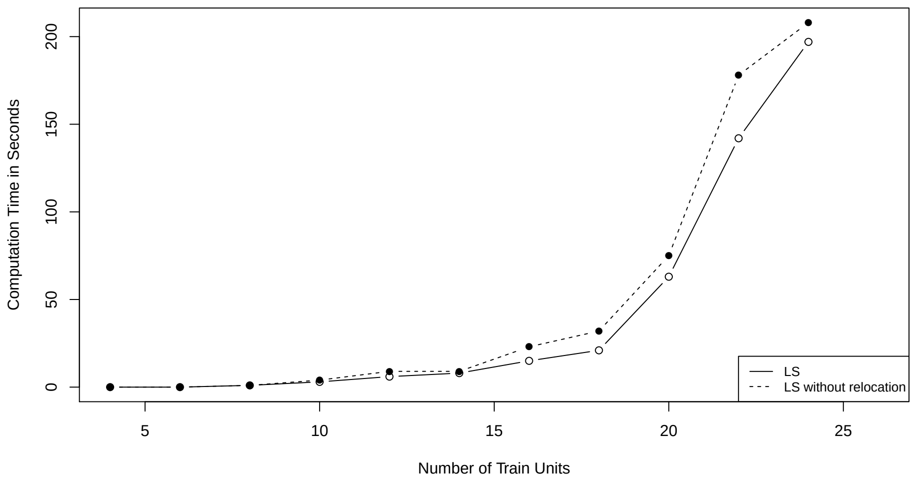


Figure 2.7: The average computation time in seconds of solved night-shift instances of the Kleine Binckhorst. The results of both the simulated annealing with the relocation operator (*LS*) and without (*LS without relocation*) are shown.

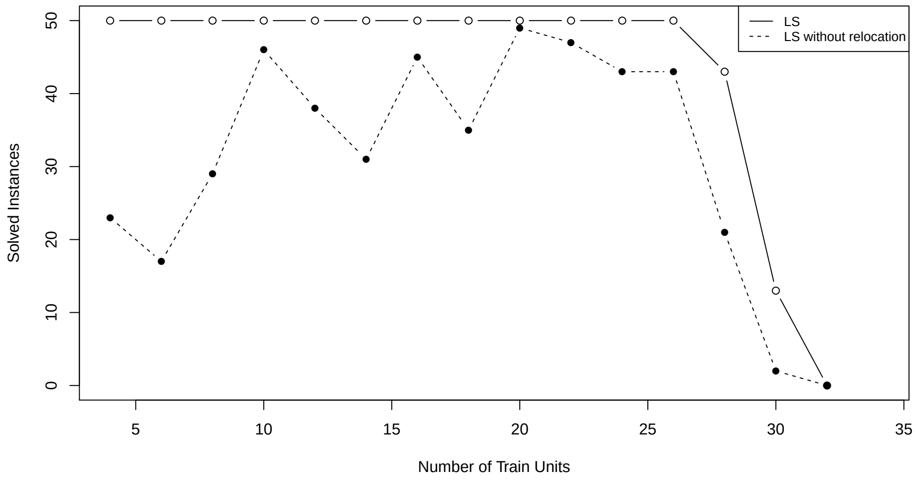


Figure 2.8: The number of feasible shunting plans found for each set of fifty night-shift instances of the Kleine Binckhorst without service tasks. The results of both the local search algorithm with the relocation operator (*LS*) and without (*LS without relocation*) are shown.

movements needed to reach the service facilities, trains will be parked on a single track for their entire stay at the shunting yard, making it difficult to resolve some conflicts. For example, suppose we have an arriving train with two coupled train units (a, b) of different train types, and a required departure composition (b, a). To reverse the composition of train (a, b) we have to split the train, move one of the two train units to the opposite side, and combine them into a single train. As splitting or combining is not allowed on the arrival track of the Kleine Binckhorst, some of the matching sub-problems can only be solved by scheduling additional train movements. In instances with a sufficient number of train units of each type, this type of conflict is often easily resolved by changing the matching. However, smaller instances are more likely to be impossible to solve without the parking relocation neighborhood, as can be seen in Figure 2.8.

In general, by removing the service tasks — using the service site only as a shunting yard — the proposed solution method is capable of finding feasible shunting plans for more train units, reaching an 85% utilization of the parking capacity of the service site in some cases. This suggests that service scheduling and the train movements to and from the facilities are a major bottleneck in the earlier experiments.

Another set of instances for the Kleine Binckhorst was generated to compare the local search algorithm with OPG, the ILP-tool developed by NS. Similar to the previous experiment, the instances do not contain service activities. However, instead of only modeling the night shift, the instances in this set span an entire day, where trains arrive in the evening and at the end of the morning, and departures occur mostly before the morning and evening rush hours. The performance of the solution methods are shown in Figure 2.9.

The differences between the two local search variants are similar to the results shown in Figure 2.8. OPG shows results resembling those of the local search without the parking relocation neighborhood, and is outperformed by the local search with relocation. So our algorithm outperforms OPG for TUSP on this type of railway yards. The similarity between the results of OPG and the local search without relocation can be explained by the mathematical model in OPG, which does not have the flexibility provided by the parking relocation neighborhood.

To test the proposed solution method for other service site layouts, we conducted similar experiments with instances generated for service site *OZ*, located near Utrecht Central Station. In contrast to the Kleine Binckhorst, most parking tracks of OZ are last-in-first-out tracks, see Figure 2.10. Trains will arrive and depart via track 117. The connection of track 117 to the main railway network prohibits parking, reversing, splitting and combining of trains on it. Parking is possible on all other tracks visible in Figure 2.10. The cleaning platform is accessible from tracks 104 and 105b. Instances for the night shift are generated using the same parameters as for the Kleine Binckhorst, with the number of train units ranging from 4 to 22. The same service tasks are assigned to the train units, with the exception of washing activities due to the absence of a washing installation. The results, shown in Figure 2.11, confirm those of the experiments with the Kleine Binckhorst, as for TUSPwSS the local search with relocation performs

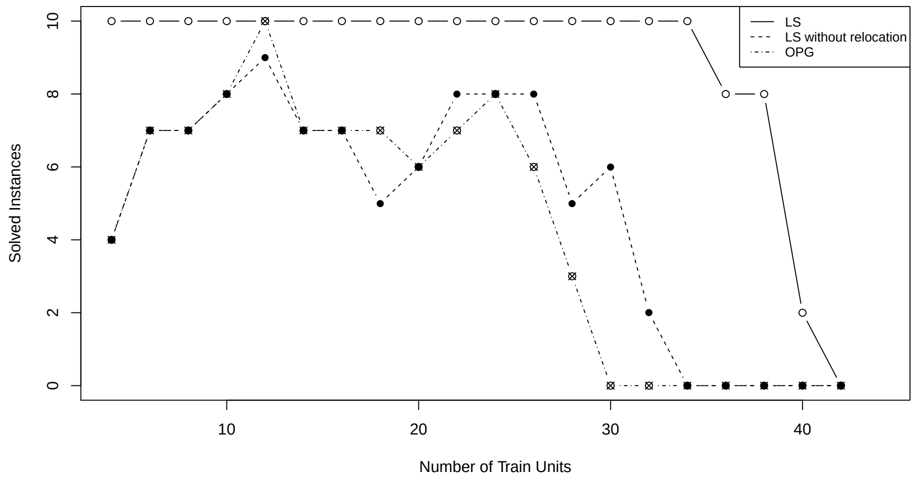


Figure 2.9: The number of feasible shunting plans found for each set of full-day instances without service tasks of the Kleine Binckhorst. The results of the simulated annealing variants with the relocation operator (LS), without relocation (*LS without relocation*) and the ILP-based OPG are shown.

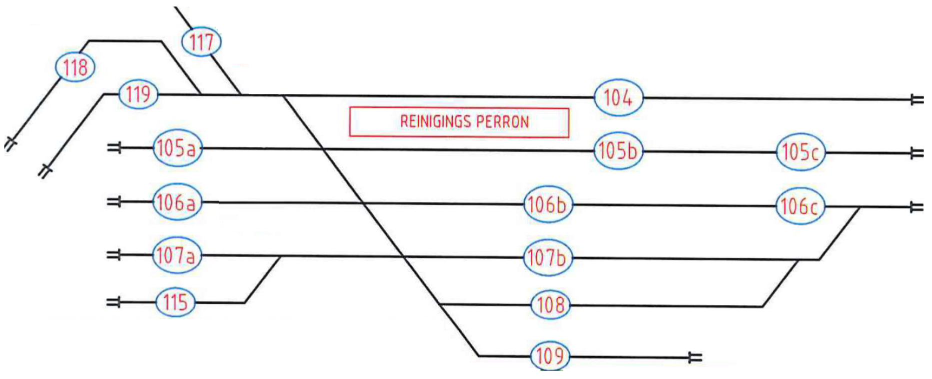


Figure 2.10: The service site *"OZ"* operated by NS.

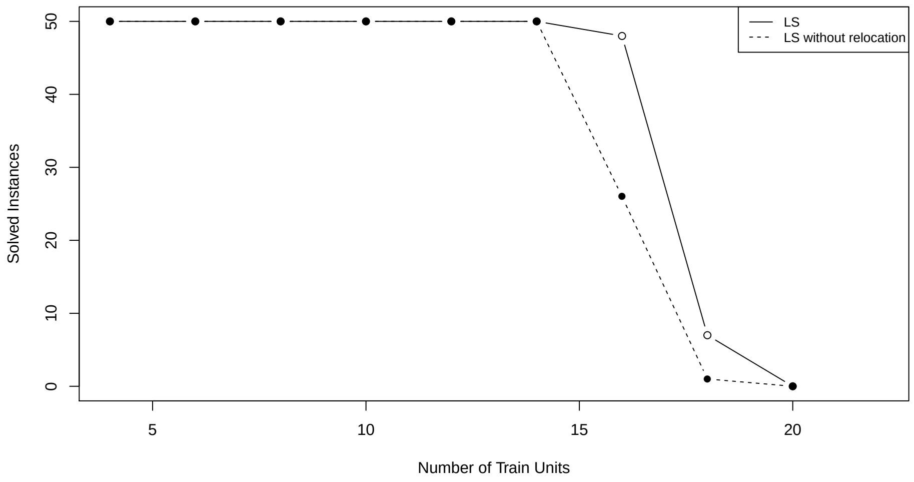


Figure 2.11: The number of feasible shunting plans found for each set of fifty instances of service site OZ. The results of both the simulated annealing with the relocation operator (*SA*) and without (*SA without relocation*) are shown.

slightly better than without the relocation operator.

In addition to the relocation neighborhood experiments we investigated the impact of the weights of the penalties listed in Table 2.5. We ran the local seach with seven different parameter settings on a subset of the instances used in the experiments in Figure 2.6. The results are listed in Table 2.9. Parameter setting 1 describes the parameter values used in the other experiments in this section.

The experiments show that penalizing the movements is necessary to solve the majority of the instances, as the parameter settings without movement penalties (settings 2 and 4) perform significantly worse than the other settings. Furthermore, a small incentive to prefer the resolution of delays over other conflicts increases the likelihood that the local search finds a feasible solution.

2.5 Integrated Station and Yard Planning

Recently, NS conducted a pilot study to construct the daily shunting plans a few days to a few hours in advance, instead of several weeks. The main benefit of planning closer to the day of implementation is that the data available to the planners is more accurate. To ensure that the shunting plans are constructed before going into operations, NS is evaluating a decision support system based on the local search algorithm developed in this thesis to help the human planners.

Two railway hubs — Heerlen and Eindhoven — were selected by NS as test

	parameter settings					results	
	w_{delay}	$w_{crossing}$	w_{track}	w_{time}	w_{move}	solved (%)	time (s)
1.	2	1	1	.00025	0.01	92	94
2.	2	1	1	.00025	0	36	182
3.	2	1	1	.00025	0.1	84	69
4.	1	1	1	0	0	40	109
5.	1	1	1	0	0.01	54	165
6.	1	1	1	.00025	0.01	48	160
7.	3	1	1	.001	0.01	66	98

Table 2.9: The percentage of solved instances and average computation time for seven sets of parameter values of the local search algorithm.

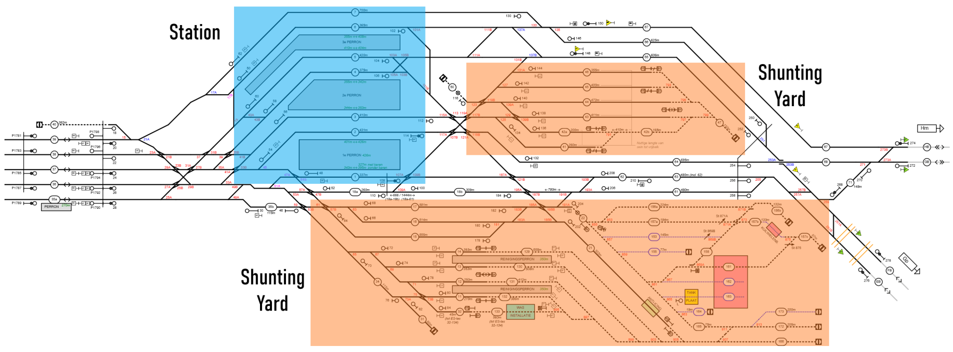


Figure 2.12: Railway hub Eindhoven. The station area is highlighted in blue, and the two orange rectangles indicate the locations of the two shunting yards.

locations for the pilot. A railway hub encompasses a major train station and one or more nearby shunting yards. Heerlen is a small hub with only a single railway yard, whereas the larger Eindhoven hub, shown in Figure 2.12, contains two yards. The goal of the decision support system in the pilot is to create integrated shunting plans for the complete hubs. The widening of the scope of the planning problem from a single shunting yard, which is the primary focus of this thesis, to a train station with multiple yards poses several new challenges to our local search algorithm. In this section we discuss some of these challenges and describe the modifications made in cooperation with NS to the local search for their pilot study. We conclude this section with an overview of the results of the pilot.

2.5.1 Network Train Movements and Out of Service Infrastructure

In contrast to shunting yards, where all train movements are scheduled by the yard planners, most of the railway traffic at train stations is predetermined by the timetable. The *network trains* are trains that move through the station without going to a railway yard, and have a fixed path, start time and duration. The movements of these network trains do not have to be planned by the local search algorithm. However, trains moving from the station to the shunting yard and vice versa do require scheduling, and these movements must not conflict with the network trains.

As a result of the movements of network trains, parts of the railway hub infrastructure are temporarily unavailable for the planning of train movements. Railway infrastructure such as tracks or switches can also be out of service to allow maintenance.

We model both the movements of network trains and the out of service infrastructure as dummy movement activities in our partial order schedule. These dummy movements have a fixed release date and deadline, and the routes of the movements represent the infrastructure that is unavailable for other train movements. The dummy movements are mostly treated by the local search algorithm as any other movement activity, i.e., precedence relations between train movements and dummy movements can be added and removed. The main differences are that the route of a dummy movement is fixed, and dummy movements cannot be removed from the shunting plan. Dummy movements in intermediate solutions of the local search might be delayed due to the precedence relations in the partial order schedule. However, these delays are penalized in the objective function and a shunting plan is only feasible when all dummy movements start at their scheduled time in the timetable.

2.5.2 Minimum Headway Times

Based on the safety regulations at shunting yards we constrained the scheduling of train movements with intersecting routes to disjoint time intervals. However, at the railway hub level these constraints are too restrictive, as movements from the train station to the shunting yard would block a significant part of the railway infrastructure for a long time. Instead, the safety regulations specify *minimum headway times* between trains moving over the same track or switch. For example, when two trains move over the same track, then the second train cannot enter the track until two minutes after the first train has left the track. The minimum headway times depend on the movement directions of the trains. While trains moving in the same direction have to keep a distance of two minutes, train movements in opposite directions can have a minimum headway time of three minutes.

For the no-overlap constraints at shunting yards it was sufficient to keep track of the route and duration of each train movement. However, to satisfy the minimum headway constraints at every track or switch, we need to store for train movements at which time the train reaches each part of the railway infrastructure along its path. Furthermore, we modify the train movement start time assignment procedure described in Section 2.3.2. Instead of simply adding a precedence relation between train movements with intersecting routes, we compute the minimum difference in the start times of the movements based on the minimum headway time on each track or switch in the intersection of the routes.

2.5.3 Day Transitions

For the sake of simplicity we assumed that shunting yards are empty at the start and the end of the planning horizon. While this assumption rarely holds in practice, we can easily model trains parked on the yard at the beginning of the planning horizon by introducing dummy arrival activities before the start of day for these trains. The trains arrive at the tracks on which they were parked during the previous day. Similarly, trains that remain on the yard at the end of the day have a dummy departure activity that can take place on any track where trains are allowed to be parked.

2.5.4 Current Results

The pilot study of NS showed promising results. The local search algorithm solves instances of regular week days at the Heerlen railway hub within 30 minutes. In these instances 70 train units would enter the station during the day, and 14 of these required maintenance of approximately two hours at the shunting yard.

The Eindhoven railway hub receives significantly more traffic than Heerlen. On a regular day the shunting plan contains 130 train units, of which 40 have to be cleaned or maintained. Furthermore, 480 movements of network trains are scheduled in the timetable. Feasible solutions could be constructed by the local search for most of the instances, and on average 230 train movements to and from the shunting yards were scheduled in the shunting plans. The computation time ranged from 30 minutes to several hours, and for some instances no feasible solution could be constructed after ten hours of computation time. These infeasible shunting plans contained only a few conflicts, and all of the conflicts could be resolved by human planners. The most challenging instance was the weekend instance, which starts at Friday evening and ends on Monday morning. With more than 3000 movements of network trains, this instance remained unsolved after 16 hours of computation time by the local search. The primary bottleneck for the local search algorithm in this instance appears to be the long planning horizon, as the number of local search iterations per time unit is significantly less for the weekend instance than with the single-day instances.

Nevertheless, even with the manual resolution of the remaining conflicts the decision support system resulted in a significant reduction of the time needed to create shunting plans for the railway hubs. Additionally, we can easily add planner preferences to the local search algorithm by adding new components to the objective functions. For example, the preference that trains of some train type are parked on a specific part of the shunting yard can be modeled by introducing small parking weights for the desired train type-track combinations. Due to the success of the pilot at the two railway hubs, NS is now starting the process of a nation-wide implementation of a decision support system based on the methods developed in this chapter.

2.6 Conclusions

In this chapter we have studied the problem of planning the parking and servicing train units at service sites operated by NS. The research is conducted with two purposes, firstly to support human planners with the construction of feasible shunting plans for the service sites, and secondly to improve the capacity estimates by the management of NS.

We have introduced the *Train Unit Shunting Problem with Service scheduling (TUSPwSS)*. Although the Train Unit Shunting Problem has been studied, there are no practical algorithms that include the resource-constrained scheduling of service tasks.

We have presented a local search approach to find feasible plans for TUSPwSS. *This is the first algorithm capable of constructing feasible plans for real-world instances of the full shunting and service scheduling problem.* The solution method consists of a plan representation that models the precedence relations between the scheduled activities, as well as local search neighborhoods exploiting the partial ordering. We have benchmarked our approach on both generated and real-world instances of service sites operated by NS. The experiments showed that our solution method is capable of solving shunting problems on service sites with varying infrastructural layouts within a few minutes.

Moreover, we compared our algorithm to OPG, a decision support tool based on state-of-the-art mathematical programming models that has been developed by NS. In the solutions of OPG trains are parked at a fixed place during their stay at the yard. OPG does not include service scheduling. Therefore, we included instances without service scheduling in our experiments.

Comparison with OPG showed that our local search algorithm is capable of solving harder instances than the ILP-based OPG. Since an important difference is the possibility for relocation, we decided to investigate this aspect further by also running our algorithm without relocation. These experiments demonstrated that the flexibility to move a train to a different track during parking is essential to find feasible plans. Note that in TUSPwSS the possibility for relocation is obtained partly from the service schedule, which forces trains to move to service facilities.

The real-world scenario illustrated that the local search approach is a valuable tool in the planning process at NS by providing human planners with feasible solutions, drastically reducing the time needed to construct good plans for service sites. The local search method is currently being used by NS to obtain a good estimate of the capacity of their service sites by generating realistic problem instances for varying numbers of train units, and checking for which number the local search algorithm can still consistently find feasible solutions.

Preliminary results from a pilot study started by NS show that in most cases we are able to find feasible shunting plans for real-world problem instances of railway nodes, which consist of a major station and one or more shunting yards. Even when the local search fails to find a feasible solution, the number of conflicts in the final solution is reduced to such an extent that human planners can resolve remaining conflicts within an hour, which is a fraction of the time that it would take them to construct a shunting plan from scratch. The pilot shows that the main strength of our local search approach lies in its flexibility, as it is easily adaptable to the often complex constraints that arise in real-world problem. This is especially appreciated by the practitioners at NS, as we are able to incorporate many of their planning preferences into our model.

Chapter 3

Surrogate Robustness Measures

3.1 Introduction

Small disruptions such as train delays occur frequently at railway yards. As these disturbances are rarely known at the time of planning, ad-hoc modifications have to be made to the shunting plans during execution. With only a few minutes to decide on a course of action, railway yard operators cannot feasibly determine the full impact of every change to the shunting plan. Consequently, the yard operator might perform a recourse action in response to a small disruption that causes major problems later in time.

A shunting plan can be described by the assignment of resources and start times to the activities on the yard. The ordering of activities assigned to the same resource is induced by the start time assignment.

To reduce the impact of small disturbances on the operational performance of the railway yards, we should construct shunting plans that are *robust*. We define the robustness of a plan as its capability of absorbing the consequences of disruptions without significant adjustments to the plan. Although during operation we do not want major adjustments, we allow a predetermined rescheduling policy that makes small plan changes.

In this chapter and Chapter 4 we assume that the rescheduling policy is a *right-shift* policy. In this policy the assignment of activities to resources as well as the ordering of activities on those resources is considered fixed during operations. The recourse actions of the yard operators are limited to reassigning the start times of activities. Yard operators might have to delay activities in response to disturbances, and the delays propagate to successor activities in the partial order schedule. The partial order schedule itself cannot be modified during operations in this rescheduling policy.

In this chapter we look at the field of stochastic scheduling problems for inspiration, since scheduling is the basis of planning at railway yard and robustness has been extensively studied in this area. In Chapter 4 we will apply the ideas on robustness obtained from the stochastic scheduling problems to the specific case of constructing robust shunting plans.

Scheduling is concerned with the allocation of tasks to scarce resources over time. The general objective of scheduling problems is to schedule tasks on resources such that some function of the completion times or resource assignment of the tasks is optimized. In deterministic scheduling problems the processing time of a task is known in advance, whereas in stochastic scheduling problems the processing times of tasks are modeled by stochastic variables. Stochastic scheduling can be viewed as a model for scheduling problems with disturbances. Disturbances in deterministic scheduling problems can be viewed as realizations in the corresponding stochastic problem, i.e., the scheduling problem where the expected processing times are the processing times from the deterministic problem and the probability distribution of the processing times models the assumed behavior of disturbances. We call a schedule robust with respect to an objective function when the degradation of that objective function caused by rescheduling in response to the stochastic disturbances is deemed acceptable. The initial schedule is called the *baseline schedule*. We usually measure degradation with respect to this baseline schedule.

In stochastic scheduling literature many characterizations of robustness have been proposed and studied. These characterizations are often categorized based on the type of properties of the schedule that should withstand disruptions. A schedule is *quality robust* if it prevents severe degradation of scheduling objectives such as makespan or lateness. Note that these objective values are random variables in stochastic scheduling problems due to the stochastic processing times of the tasks. Examples of quality robust schedules are schedules that minimize the average makespan.

When disturbances occur, a *solution robust* schedule minimizes the variance of some properties of the schedule with respect to the baseline schedule. For example, in timetabling problems the objective is to publish a timetable, which is an assignment of start times to activities, such that disturbances do not cause large deviations from the published start times.

Other differences in robustness characterizations stem from the assumptions on the type and severity of the disturbances. Schedules might be robust to small deviations in the processing times of tasks, yet deteriorate strongly when unplanned tasks have to be inserted in the existing schedule.

After settling on a specific characterization of robustness, the practical question of "how to quantify the robustness of a schedule?" remains. Exact analytical expressions of most robustness characterizations can be computationally hard to evaluate in solutions to stochastic scheduling problems, due to the interdependencies of the stochastic variables.

An approach often used to estimate robustness is to simulate the schedule in many possible scenarios sampled from the (assumed) distributions of the uncertainty. Simulation is a powerful and versatile tool that gives an accurate estimate of a robustness characterization if a sufficient number of samples is used, but it tends to be a computationally expensive technique. Hence, simulation is very useful to compute the robustness of a given schedule, but it is hard to include when searching for a highly robust solution. As solution methods for scheduling problems typically evaluate a large number of schedules to find a (near-)optimal solution, using simulation as a subroutine in a solution method might not be feasible.

An alternative method to estimate the robustness of a schedule is to look at the characteristics of the schedule in a simplified stochastic model. These characteristics are known as *(surrogate) robustness measures.* Due to their computational efficiency many robustness measures have been proposed in literature. However, the efficacy of these surrogate robustness measures as predictors of robustness strongly depends on the specific characterization of the robustness as well as the characteristics of the scheduling problem.

The goal of this chapter is to provide an overview of robustness measures for stochastic scheduling problems and to study the interaction of the surrogate robustness measures with several characteristics of robustness. In Section 3.2 we discuss several characterizations of robustness. We continue in Section 3.3 with an overview of literature on robustness measures, and describe several of these robustness measures in more detail in Section 3.4. We conclude this chapter with several example schedules to illustrate the properties of the robustness measures in Section 3.5.

3.2 Preliminaries of Stochastic Scheduling

In the deterministic problem of parallel machine scheduling we are given a set of n tasks (activities) and m identical machines (resources) to process these tasks. Each task has to be processed by one of the machines. The processing time of task i is denoted as p_i. Each machine can only process a single task at a time, and once a machine starts processing a task it must completely process that task before it can start with a different task.

A solution to the parallel machine scheduling problem, which we will refer to as a *schedule*, is a feasible assignment of machines and start times to the tasks. The standard objective of parallel machine scheduling is to find a schedule that minimizes the *makespan*, which is the maximum completion time over all tasks.

In machine scheduling problems the feasible starting times of a task i are often constrained by a *release date* r_i, which is a lower bound on the starting time, and/or a *deadline* $\bar{d}_i$, which is an upper bound on the maximum completion time of the task. Furthermore, *precedence constraints* on the tasks impose a partial ordering of those tasks in time. That is, a precedence constraint $i \prec j$ indicates that task i must be completed before the start time of task j.

We can generalize the parallel machine scheduling problem to a *resource-constrained project scheduling problem.* Here we have multiple types of resources to execute the tasks. Each task requires a given amount of certain resources, where a task may require different resources in parallel. For example, the repair of a car may take 2 hours of a repairman and need a lift bridge and assistance of another repairman during the first hour. In the remainder of this section we assume that each task requires a single type of resource and that each resource can process one task at a time. Furthermore, resources of the same type are identical. This boils down to introducing multiple types of parallel machines and adding to each task the constraint that it is processed by a machine of a specific type.

Solving a resource-constrained scheduling problem consists of three elements, namely

1. assigning the tasks to the resources;
2. ordering tasks assigned to a resource;
3. assigning start times to the tasks.

A *partial order schedule* ($\mathcal{POS}$) is a set of precedence relations of the tasks that solves these first two elements. The precedence relations in $\mathcal{POS}$ either constrain the processing order of tasks assigned to the same resource or enforce the precedence constraints in the scheduling problem. The partial order schedule does not explicitly assign the tasks to the resources. However, assigning tasks that require the same resource type and are ordered in the partial order schedule to the same resources constructs a valid task-resource assignment.

We can extend a partial order schedule to a schedule that covers all three elements of a resource-constrained scheduling problem by assigning start times to the tasks such that the start time of each task is at least the maximum completion time of its predecessors in the partial ordering. Note that a partial order schedule can be extended to many start time assignments. From the $\mathcal{POS}$ we can compute for each task i the time window in which it has to be processed.

The *earliest start time* est_i is the earliest possible time at which all predecessor tasks of i can be finished. The *earliest completion time* ect_i is obtained by adding the processing time p_i to the earliest start time of i. The maximum earliest completion time over all tasks is the *makespan* of the schedule, and is denoted as C_{max}. A *critical path* is a sequence of tasks in the partial ordering that cannot be completed before C_{max}.

The end of the time window in which task i has to be processed is denoted by lct_i, which is the *latest completion time* of i. The latest completion time is equal to the latest possible completion time of task i such that the schedule remains feasible with respect to the deadline constraints. In the absence of deadline constraints we assume a global deadline for all tasks that is equal to the makespan C_{max} of the schedule. The *latest start time* lst_i is the latest completion time subtracted by the processing time of task i.

The robustness of a schedule is strongly related to the amount of time by which we can delay the start of a job without suffering serious consequences. In the literature such a delay is commonly referred to as *slack*. Many types of slack have been considered.

We define the *total slack* ts_i of task i in the partial ordering as the maximum amount of time by which we can delay the task such that no deadlines are exceeded in the schedule. Equivalently, the slack is the difference between the earliest and latest start time of the task, $ts_i = lst_i - est_i$. Note that in the absence of deadline constraints the total slack of tasks on a critical path of the schedule is equal to 0.

A different type of slack is the *free slack* fs_i, which is the maximum amount of time that task i can be delayed without affecting any other task. That is, we define the free slack as

$$fs_i = \min_{j \in succ(i)} \{est_j\} - ect_i, \tag{3.1}$$

where $succ(i)$ are the successor tasks of i in the schedule. The free slack of a task i is never larger than the total slack of i.

Since the input of a stochastic scheduling problem contains random variables, most properties of a solution to the problem will be stochastic as well. In particular, an objective function such as the makespan of the schedule can no longer be expressed as a single deterministic value. To compare solutions in stochastic scheduling problems a common approach is to restrict the objective functions to deterministic properties of the random variables. Examples of such objectives are minimization of the expected value or the variance of the schedules.

In stochastic scheduling problems with deadline constraints the feasibility of a schedule cannot be guaranteed in most cases, since the completion times of the tasks are random variables. Instead of determining the absolute feasibility of a solution, we can compute the likelihood that the deadlines are not exceeded. In many real-life scheduling problems the likelihood of feasibility of a schedule is either maximized in the objective function or constrained by a lower bound to ensure that the schedules can be applied in practice.

3.3 Literature Overview

Most of the robustness measures proposed in literature are for resource-constrained project scheduling problems, where the standard objective is to minimize the makespan of the schedule. These measures are mainly based on the concepts of total slack and free slack.

A simple slack-based robustness estimation, proposed by Jorge Leon et al. (1994), is to compute the average of the total slacks of all activities. By simulating many realizations of job shop schedules, Jorge Leon et al. (1994) showed that a large percentage of the variation in the realized makespan was explained by the average slack of the schedule. Similarly, Al-Fawzan et al. (2005) proposed the sum of free slacks as a robustness measure.

Based on the observation that, in addition to the total amount of slack, the distribution of the slack over the schedule affects the robustness as well, Chtourou et al. (2008) proposed several variants of the sum of free slacks. These robustness measures weigh the free slack by the number of successors, or substitute the free slack with a binary slack indicator function or an upper bound on the slack based on the activity duration.

The relation of a number of existing and newly proposed robustness measures to the fraction of feasible schedule realizations in a Monte Carlo simulation has been investigated by Hazır et al. (2010). For instances of the discrete time/cost trade-off problem, they reported high values (> 0.91) of the coefficient of determination for the sum of total slacks measure and successor-weighted variants of it.

A similar comparison of robustness metrics in a Monte Carlo simulation was performed by Canon et al. (2010). In contrast to the work of Hazır et al. (2010), their results showed that summing the unweighted slack of the activities has at best a weak correlation with the expected makespan of the schedule.

When scheduling activities subject to deadlines, the primary objective is to find a feasible schedule. However, the concepts of free and total slack do not fully capture the slack of a schedule with respect to its deadline. To quantify this type of slack, we can view a schedule with deadlines as a special case of a *Simple Temporal Network (STN)*, which is a directed graph with both minimum and maximum time lags on the arcs that was introduced by Dechter et al. (1991). Similarly to the slack for schedules, several *flexibility metrics* have been proposed for this type of graph, which aggregate the slack with respect to all the temporal constraints, including the deadlines. The *naive flexibility* of an STN is the sum of the difference between the latest and earliest start time of each activity, i.e., the total slack relative to the deadline instead of the makespan of the schedule. Analog to the free slack of an activity, Wilson et al. (2014) proposed the *concurrent flexibility* metric, which is based on interval schedules. An interval schedule specifies for each activity an interval such that every activity can start at any time within its interval independently of the other events, and without exceeding the deadline of the schedule. The concurrent flexibility of an STN is defined as the maximal sum of the interval lengths over all possible interval schedules. A linear programming formulation was proposed by the authors to compute the concurrent flexibility. It was shown in Wilson (2016) that a schedule with a high flexibility is not always robust to disruptions.

The limitations of the sum of free slacks metric were discussed by Kobylański et al. (2007), and they proposed to use the minimum free slack over all activities as a robustness measure for schedules with a deadline, and provided an algorithm that maximizes the minimum free slack by distributing the free slack evenly over the schedule. Their approach is essentially the concurrent flexibility metric, proposed by Wilson et al. (2014), with the objective to maximize the minimum interval length instead of the sum of the intervals.

An extensive comparison of robustness measures can be found in the paper of Khemakhem et al. (2013). They investigated the correlation between the surrogate robustness measures and the probability that the completion time of a schedule exceeds its nominal makespan, which was approximated using a Monte Carlo simulation. Their results showed that the strongest correlation ($R^2 > 0.64$) with the robustness performance metric in the simulation was achieved with a robustness measure that computes the *slack sufficiency*, which is based on the ratio between the free slack and the processing time of an activity.

Despite the many surrogate robustness measures in literature, there is no consensus on which of these provides a good approximation of the true robustness of a schedule. In the simulation studies of Jorge Leon et al. (1994), Hazır et al.

(2010), Canon et al. (2010) and Khemakhem et al. (2013), only schedules without deadlines are considered, focusing mainly on the expected makespan and related performance metrics. However, a good expected makespan of a schedule constrained by a deadline does not necessarily imply that the schedule will respect its deadline. Therefore, we verify their results for schedules with deadlines in Chapters 3 and 4.

3.4 Robustness Measures

Surrogate robustness measures that assume that the exact probability distribution of the uncertain data is known might produce accurate predictions of the robustness, but their applicability to real-world scheduling problems is limited, since quantitative data of the uncertainty are often scarce in practice. Therefore, robustness measures with a low dependency on the available knowledge of the uncertainty are preferred.

Robustness measures are usually created with the assumption that the mean processing times of the activity are known. If a robustness measure does not rely on any other information about the uncertainty, the robustness is solely estimated from the structure of the partial order schedule σ. The two robustness measures applied most often in literature, namely the sum of total slacks (Jorge Leon et al. (1994)),

$$RM_1(\sigma) = \sum_i ts_i^\sigma, \tag{3.2}$$

and the sum of free slacks (Al-Fawzan et al. (2005)),

$$RM_2(\sigma) = \sum_i fs_i^\sigma, \tag{3.3}$$

are examples of measures that depend only on the mean activity durations. When all activities are subject to deadlines, then the minimum total slack,

$$RM_3(\sigma) = \min_i ts_i^\sigma, \tag{3.4}$$

reflects the slack between the completion time of a critical path and its deadline. In that case, the minimum total slack can be viewed as a robustness measure as well. Recall that in a minimum makespan schedule the minimum total slack equals zero by definition.

Another robustness measure of this type that was shown by Khemakhem et al. (2013) to provide good estimations of the robustness of a schedule was based on *slack sufficiency*, which compares the free slack of an activity to a fraction of the duration of that activity or one of its predecessors in the schedule. In the work of Khemakhem et al. (2013), this robustness measure is defined as

$$RM_4(\sigma, \lambda) = \sum_i |\{j \mid j \in prec_\sigma(i) \cup \{i\}, fs_i \geq \lambda p_j\}| \tag{3.5}$$

where $prec_\sigma(i)$ are the predecessors of activity i in the schedule and $0 < \lambda < 1$. The authors suggested that λ should be set to the expected deviation from the nominal processing times of the activities due to disruptions.

A more complex robustness measure depending only on the expected activity durations is the interval schedule based approach of Wilson et al. (2014). Given a partial ordering of the activities and for each activity i an interval (est_i, lst_i) in which i must start, the approach finds a maximal assignment of intervals to activities such that each activity i can be scheduled within its interval (e_i, l_i) *independently of the other activities.* To achieve this, the intervals are computed with the linear program

$$\begin{aligned} RM_5(\sigma) = & \max \sum_i (l_i - e_i) \\ & \text{subject to} \\ & \quad est_i^\sigma \le e_i \le l_i \le lst_i^\sigma \quad && \forall i \\ & \quad l_i + p_i \le e_j && \forall i \prec j \in \mathcal{POS}_\sigma. \end{aligned} \tag{3.6}$$

As an alternative to solving this linear program, Mountakis et al. (2015) formulated a matching problem based on the dual problem.

Analog to the work of Kobylański et al. (2007), we can change the objective of the linear program of Wilson et al. (2014) to maximize the minimum interval, which will result in a more evenly distributed interval schedule. The linear program then becomes

$$\begin{aligned} RM_6(\sigma) = & \max \min_i (l_i - e_i) \\ & \text{subject to} \\ & \quad est_i^\sigma \le e_i \le l_i \le lst_i^\sigma \quad && \forall i \\ & \quad l_i + p_i \le e_j && \forall i \prec j \in \mathcal{POS}_\sigma. \end{aligned} \tag{3.7}$$

In contrast to the previous surrogate robustness measures, the measures RM_5 and RM_6 focus on the entire graph structure instead of just the slack of the individual activities. However, an optimal solution to either of the linear programs is an interval schedule that assigns large intervals to concurrent activities, and, consequently, only small intervals to sequential activities, thus overemphasizing parallel activities.

A compromise between activity-based and schedule-based measures is to predict the robustness of a schedule from the paths in the partial ordering. We define the slack of a path of tasks $\pi = (\pi_1, \ldots, \pi_l)$ in the schedule σ as

$$s_\pi^\sigma = lct_{\pi_l}^\sigma - est_{\pi_1}^\sigma - \sum_{i \in \pi} p_i.$$

Without any knowledge of the uncertainty, it is reasonable to assume that the likelihood of a disruption on a path increases with the number of activities of the path. Therefore, we propose to use the minimum over all paths of the path slack divided by the number of activities on the path as a robustness measure,

$$RM_7(\sigma) = \min_{\pi} \left\{ \frac{s_\pi^\sigma}{|\pi|} \right\}. \tag{3.8}$$

Although the number of paths can be exponentially large, we can evaluate this robustness measure efficiently by computing for each $k = 1$ to $n + 2$ the shortest path lengths in the schedule with exactly k activities.

In many cases, a reasonable estimate of the variance of the uncertainty can be made as well, even if the exact distribution of the uncertainty is unknown. We can exploit this additional information by making the assumption that the duration of each activity is normally distributed, as normal distributions can be characterized solely by their mean and variance. Although this assumption might be wrong for the distribution of the duration of a single activity, if follows from the central limit theorem that the sum of activity durations does resemble a normal distribution. Therefore, we can approximate the uncertainty in the duration of a path in the schedule.

We can utilize this approximation as the basis for several robustness measures. Firstly, we propose another path-based robustness measure. Analog to the minimum weighted path slack in RM_7, we use the minimum probability that a path can be completed within the deadline, computed over all the possible paths in the graph. That is, we compute for each path π the normal distribution approximation X_π of the duration of the path by summing the processing time distributions of activities on that path, and report the minimum probability of completion before the deadline T:

$$RM_8(\sigma) = \min_{\pi} \{P(X_\pi \leq T)\} \tag{3.9}$$

Although the paths in the schedule are connected by precedence relations, they are assumed to be independent by this robustness measure.

In contrast to RM_7, we might have to evaluate all the paths in the schedule to compute the distribution-based robustness measure RM_8, since the usual graph-theoretical properties of paths, such as the property that any sub-path of a shortest path is again a shortest path, do not hold in this case. To keep the computation tractable, we construct in topological order for each activity i the set of paths in schedule σ ending at i, from the paths ending at the immediate predecessors of i:

$$\Pi_i = \left\{ (\pi^1, \ldots, \pi^k, i) \mid \exists\, j \prec i \in \mathcal{POS}_\sigma : (\pi^1, \ldots, \pi^k) \in \Pi_j \right\}. \tag{3.10}$$

Furthermore, we compute the distribution X_π of the duration of each $\pi \in \Pi_i$ by summing the normal distributions of the activities on the path. We can then reformulate RM_8 to

$$RM_8(\sigma) = \min_{\pi \in \Pi_{n+1}} P(X_\pi \leq T). \tag{3.11}$$

To avoid the exponential growth of Π_i, we repeatedly remove the path π from Π_i with the smallest probability of exceeding any other path in Π_i, until the set of paths is at most size K. Ties are broken randomly in this pruning procedure, and a maximum size of $K = 8$ was shown to be sufficiently large to achieve a good robustness estimation in preliminary experiments.

Another approach is to estimate the distribution of makespan of the schedule. Many approximation algorithms have been proposed in literature, see Ludwig et al. (2001) for a comparison of several techniques. An efficient method to construct an approximation of the makespan distribution is to evaluate the activities in topological order, computing the makespan distribution Y_i up to each activity i as the distribution of the maximum over the makespan distributions of the immediate predecessors of i

$$Y_i = \max_{j \prec i \in \mathcal{POS}_\sigma} \{Y_j\} + D_i, \tag{3.12}$$

where D_i is the normal approximation of the activity duration of i, and the maximum over the predecessor distributions is approximated with a normal distribution as proposed by Nadarajah et al. (2008). The robustness measure, which is proposed in the work of Passage et al. (2016), is then

$$RM_9(\sigma) = P\left(Y_{n+1} \leq T\right). \tag{3.13}$$

3.5 Examples

Although the goal is finding robust shunting plans, we will look at four small examples of parallel machine schedules with deadlines and stochastic programming times to illustrate the behavior of the robustness measures. In each of the schedules we have 16 tasks that are assigned to 4 parallel machines, with 4 tasks per machines. Figure 3.1 shows the four schedules.

In a simulation study we evaluated the performance of the schedules with different processing time distributions and deadlines. The processing times of the tasks are sampled from either a normal distribution, a log-normal distribution or exponential distribution. The mean processing time of each task is 10. The standard deviation σ of the normal and log-normal distributions is set to 3, and the σ of an exponential distribution is equal to its mean.

The deterministic makespan of the schedules is 40. We apply a global deadline constraint to each schedule of either 50 or 60 time units.

The performances of the schedules for the different parameter settings are presented in Table 3.1. The results show that the additional precedence constraints cause more delays in the schedules, especially when the standard deviation of the processing times is high.

We compare the simulation results with some of the robustness measures discussed in this chapter. The values of the robustness measures are listed in Table 3.2. Although the slack-based robustness measures do show the effect of extending the deadline, most of them fail to capture the impact of the additional precedence constraints. Furthermore, the slack-based measure cannot differentiate between the processing time distributions, as they only use knowledge on the mean processing time. In contrast, the values of the normal approximation

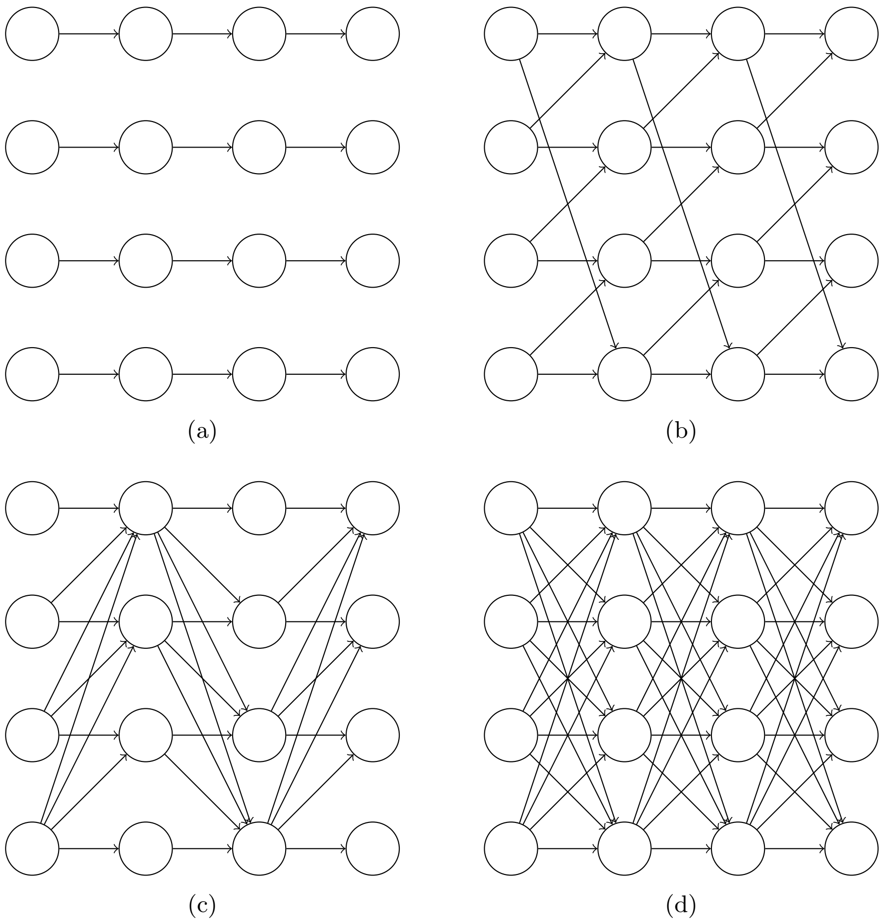


Figure 3.1: The four tasks in each row of a schedule are assigned in order to the same machine. The arcs indicate the precedence constraints in the schedule.

		average C_{max}			fraction within deadline		
schedule	deadline	normal	log-normal	exponential	normal	log-normal	exponential
3.1a	50	46.4	46.5	62.0	0.82	0.78	0.29
3.1a	60	46.4	46.5	62.0	0.99	0.99	0.50
3.1b	50	49.9	50.2	73.6	0.52	0.52	0.11
3.1b	60	49.9	50.2	73.6	0.99	0.96	0.48
3.1c	50	50.8	51.3	77.3	0.44	0.43	0.09
3.1c	60	50.8	51.3	77.3	0.98	0.94	0.24
3.1d	50	52.5	53.2	83.6	0.27	0.29	0.05
3.1d	60	52.5	53.2	83.6	0.96	0.89	0.16

Table 3.1: The average performances of the schedules in simulations with 1000 samples.

schedule	*sd*	deadline	RM_1	RM_2	RM_3	RM_4	RM_5	RM_6	RM_7	RM_8	RM_9
3.1a	3	50	160	40	10	16	40	2.5	2.5	0.95	0.83
3.1a	3	60	320	80	20	16	80	5	5	0.99	0.99
3.1a	10	50	160	40	10	16	40	2.5	2.5	0.69	0.11
3.1a	10	60	320	80	20	16	80	5	5	0.84	0.40
3.1b	3	50	160	40	10	40	40	2.5	2.5	0.95	0.35
3.1b	3	60	320	80	20	40	80	5	5	0.99	0.99
3.1b	10	50	160	40	10	40	40	2.5	2.5	0.69	0.01
3.1b	10	60	320	80	20	40	80	5	5	0.84	0.03
3.1c	3	50	160	40	10	46	40	2.5	2.5	0.95	0.16
3.1c	3	60	320	80	20	46	80	5	5	0.99	0.99
3.1c	10	50	160	40	10	46	40	2.5	2.5	0.69	0.00
3.1c	10	60	320	80	20	46	80	5	5	0.84	0.01
3.1d	3	50	160	40	10	52	40	2.5	2.5	0.95	0.03
3.1d	3	60	320	80	20	52	80	5	5	0.99	0.96
3.1d	10	50	160	40	10	52	40	2.5	2.5	0.69	0.00
3.1d	10	60	320	80	20	52	80	5	5	0.84	0.01

Table 3.2: The estimates of the robustness measures of the different schedules and parameter settings. *sd* is the standard deviation of the processing time distribution.

method RM_9 reflect the precedence constraints in the schedules and deadlines and standard deviations parameters of the scheduling problem.

In the next chapter we will study the robustness measures in the context of the train unit shunting problem with the aim of constructing more robust shunting plans.

Chapter 4

Application of Robustness Measures to Shunting

4.1 Introduction

In Chapter 3 we have discussed the theoretical aspects of robustness measures. In this chapter we return to our practical problem of finding shunting plans that are robust to small disturbances in real-life railway yard operations. There are multiple sources of uncertainty that contribute to these disturbances. Disruptions on the main railway network can result in delayed or unscheduled train arrivals at the yard. Furthermore, service tasks and train movements often have slight deviations in their processing times. In this chapter we focus on mitigating the small disturbances that occur frequently during the execution of a shunting plan. Therefore, we limit the uncertainty to stochastic train arrivals and processing times of tasks.

The goal of this chapter is constructing robust shunting plans for the Train Unit Shunting Problem. We define the robustness of a shunting plan as the likelihood that all trains depart at their scheduled times in the time table. I.e., a robust shunting plan has a high probability that it can be implemented in practice without conflicts or delayed train departures. Shunting plans are adapted to disturbances by applying the right-shift rescheduling policy described in Chapter 3. The partial order schedule of a shunting plan is assumed to be fixed during operations to prevent conflicts in the assignment of the activities to the resources.

In order to achieve the goal of this chapter we first identify a set of robustness measures that correlate strongly to the robustness of shunting plans in a simulation study in Section 4.2. Then, we investigate in Section 4.3 whether these robustness measure can be used to guide the local search described in Chapter 2 to robust solutions. We conclude this chapter with directions of further research on reactive scheduling in Section 4.4.

4.2 Identifying Predictive Robustness Measures

The main application of the robustness measures described in Chapter 3 is in the comparison of schedules, since these measures can often be computed far more efficiently than other approaches such as simulation. However, we need to investigate whether the estimations correctly reflect the relative ordering of schedules according to their robustness to verify that the robustness estimators are actually suitable for this purpose. To accomplish this, we construct empirical makespan distributions of a set of realistic schedules in a simulation study, and search for robustness measures that show a strong correlation with the empirical results.

We have selected two real-world instances of the train unit shunting problem as the basis of our simulation study. The first one originates from *"Kleine Binkchorst (KBH)"*, which is a shunting yard near the central station of The Hague. It consists of a single night during which 19 train units arrive at the yard. These train units need to receive internal cleaning and a maintenance inspection; three of them need to be washed as well. Due to all necessary train movements, shunting plans of this problem instance typically have close to 160 activities, with 250 to 300 precedence relations. The other instance is obtained from a shunting yard near Utrecht, named *"OZ"*, which contains, contrary to the KBH, many dead-end tracks. As a result, the main difficulty in the scheduling problem is the parking order of the trains. This instance has 16 train units and a total of 27 service activities. The number of activities in the corresponding shunting plans ranges from 140 to 160 activities, and roughly 300 precedence relations.

For each of these two scheduling problems, we generated 500 feasible shunting plans with the local search method described in Chapter 2. The main components of the uncertainty in the execution of a shunting plan are the arrival times of trains and the durations of service activities and train movements. Disturbances in the arrival time of a train are modeled with a uniform distribution with the mean equal to the scheduled arrival time, and an interval size of 10 minutes. The service activities and train movements always have a nonnegative duration, and the sizes of the disruptions are usually proportional to the duration of the activities. Therefore, we model the uncertainty in these activities with a log-normal distribution with the nominal duration as the mean, and a standard deviation equal to 0.1 times the nominal duration. Robustness measures RM_1 to RM_7 use only the nominal durations in their computations, while RM_8 and RM_9 require the standard deviation of the distributions as well. Although for RM_4, the slack sufficiency measure, we can pick any value between zero and one for the fraction λ, we set it equal to the standard deviation of the uncertainty of the service and movement activities, i.e., $\lambda = 0.1$, as is suggested in Khemakhem et al. (2013).

The schedules are then evaluated by each of the robustness measures listed in Section 3.4 to generate their predictions of the robustness of the schedules. The predictions are compared with an estimate of the robustness computed using Monte Carlo simulation, which is a technique that repeatedly draws samples from the distribution of the uncertainty to simulate different realizations of the scheduling problem. To obtain an accurate estimate of the robustness, we collect 20000

	Fraction delayed		**Average delay**		**Computation**
	ρ	r	ρ	r	**Time (ms)**
RM_1	-0.840	-0.663	-0.840	-0.828	0.01
RM_2	0.456	0.357	0.470	0.491	0.02
RM_3	-0.955	-0.838	-0.972	-0.990	0.01
RM_4	0.457	0.290	0.479	0.507	0.54
RM_5	-0.298	-0.293	-0.321	-0.326	3.95
RM_6	-0.964	-0.718	-0.955	-0.889	6.64
RM_7	-0.963	-0.719	-0.953	-0.887	0.87
RM_8	-0.982	-0.969	-0.972	-0.835	0.57
RM_9	-0.981	-0.971	-0.971	-0.846	0.23

Table 4.1: The Spearman (ρ) and Pearson (r) correlation coefficients, as well as the average computation time, for the KBH instances. Coefficients close to -1 indicate that the robustness measure is a good approximation of the schedule robustness.

samples per schedule.

We use two different estimates of the robustness. Since the objective of the railway yard planners is to find feasible shunting plans that minimize the probability of delayed departures, we use the fraction of samples in which the schedule realization resulted in one or more delayed train departures as a performance metric. Additionally, we compute the average train departure delay to get a better understanding of the problem structure.

The correlation between the performance metrics and the robustness measures is investigated by computing both the *Pearson correlation coefficient* (r), and *Spearman's rank correlation coefficient* (ρ). If the robustness of a schedule can be approximated by a robustness measure, then a high value of the measure should indicate a low delayed fraction and average departure delay, and we expect that the robustness measure will have a correlation coefficient close to -1 with either of the performance metrics. A coefficient of -1 for the Pearson correlation means that there is a perfect linear relation between the robustness measure and the performance metric, and a robustness measure with a Spearman correlation of -1 will rank the schedules perfectly according to the performance metric. The Spearman correlation coefficient is particularly suitable for our experiments, since the purpose of the robustness measures is to compare schedules efficiently.

In addition to the two robustness performance metrics, we record the time required by each robustness measure to evaluate the schedules to compare the computational efficiency of the measures. To obtain reliable estimates of these computation times, we evaluated each of the 500 schedules 100 times with every robustness measure, and compute the average computation time per evaluation.

The relations between the robustness measures and the fraction of samples in which trains departed with a delay in the simulation study are shown in Figure 4.1

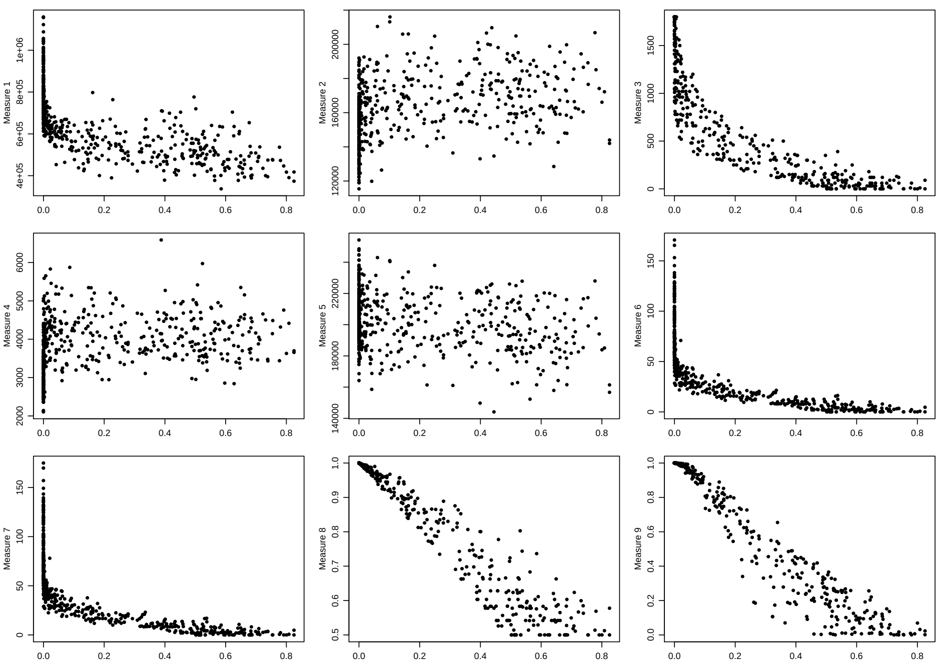


Figure 4.1: Scatter plots for the 9 robustness measures, showing the computed value of the measure (vertical axis) and the fraction of delayed samples (horizontal axis) of the KBH instances.

	Fraction delayed		**Average delay**		**Computation**
	ρ	r	ρ	r	**Time (ms)**
RM_1	-0.933	-0.831	-0.943	-0.962	0.01
RM_2	0.544	0.510	0.542	0.606	0.01
RM_3	-0.975	-0.876	-0.980	-0.989	0.01
RM_4	0.156	0.132	0.161	0.270	0.53
RM_5	-0.118	-0.117	-0.127	-0.151	3.81
RM_6	-0.969	-0.840	-0.976	-0.972	6.28
RM_7	-0.968	-0.841	-0.975	-0.972	0.83
RM_8	-0.977	-0.959	-0.971	-0.818	0.60
RM_9	-0.972	-0.910	-0.960	-0.896	0.25

Table 4.2: The Spearman (ρ) and Pearson (r) correlation coefficients, as well as the average computation time, for the OZ instances. Coefficients close to -1 indicate that the robustness measure is a good approximation of the schedule robustness.

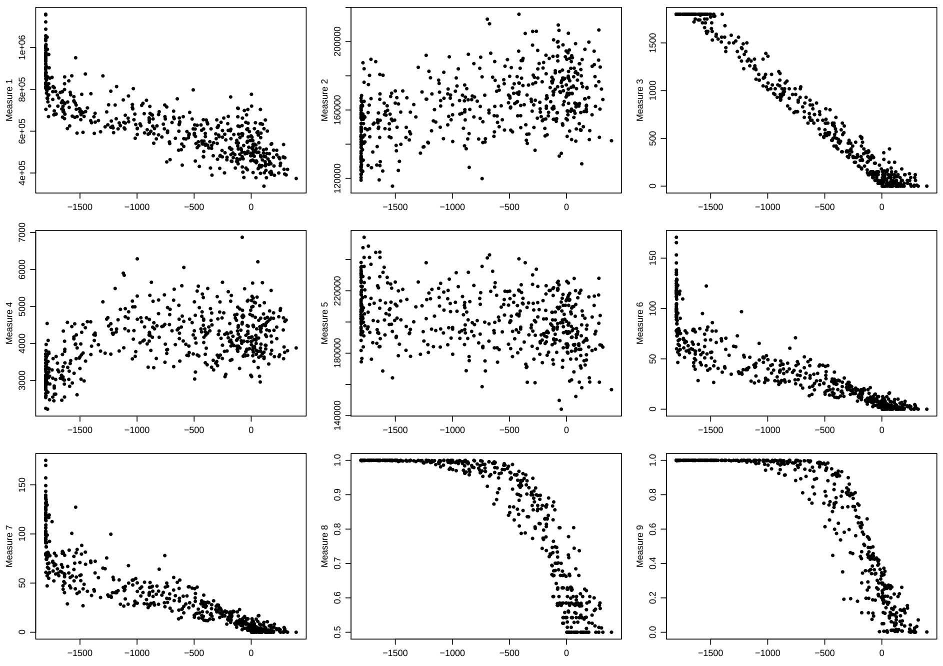


Figure 4.2: Scatter plots for the 9 robustness measures, showing the computed value of the measure (vertical axis) and the average departure delay over the samples (horizontal axis) of the KBH instances.

for the *Kleine Binckhorst* test set. Figure 4.2 shows the results for the average departure delay performance metric for the same instances. Tables 4.1 and 4.2 list the correlation coefficients of the robustness measures and the two performance metrics, as well as the average computation time, of the KBH and OZ instances.

The two robustness measures based on normal approximations, RM_8 and RM_9, appear to have the strongest rank correlation with both the performance metrics, clearly showing the advantage of exploiting the additional information of the variance of the uncertainty. Furthermore, both measures show a high Pearson correlation coefficient with the delayed fraction metric, as can be seen in Figure 4.1. If we take the computation time into account as well, then the approximation of the completion time distribution, RM_9, would be the preferred robustness measure in a practical application.

When knowledge of the variance is not available, robustness measures that rely only on the nominal processing time of activities have to be used. Of those measures, RM_3, RM_6 and RM_7 are good choices in practice due to their high correlation with both performance metrics. In particular, the minimum total slack RM_3 shows a strong Spearman correlation with the robustness performance metrics, and the correlation appears to be linear with the average departure delay metric. Given that the slack measures can be computed more efficiently than the normal approximation methods, this robustness measure will most likely be sufficient to obtain robust solutions to scheduling problems with deadlines.

Contrary to the result of Khemakhem et al. (2013), the robustness measure RM_2, RM_4 and RM_5, which are based on maximizing the sum of the free slacks, correlate poorly to either of the performance metrics. This result is supported by the random scattering of the three measures in Figures 4.1 and 4.2. In the case of RM_2 and RM_4, the probability of delays in the schedule actually increases with the total amount of free slack in the schedule. Although the cause of this relation remains to be investigated, one possible explanation might be that free slack in these shunting plans mostly arises when a train is scheduled to wait until a route or a resource is available for its movement or service activity. Therefore, if a shunting plan contains many waiting trains, then the infrastructure or resources at the shunting yard are not used effectively.

4.3 Constructing Robust Shunting Plans

The objective function of the local search proposed in Chapter 2 focuses on the conflicts in the shunting plan; the goal is to minimize the penalties incurred due to violations of the hard constraints. Since the objective is limited to the feasibility of the shunting plan, it will likely lead to solutions that handle disruptions poorly. One approach to steer the local search towards more robust solutions is to include a robustness measure in the objective function.

Based on the results in the previous section, we have selected three robustness measures that approximate the delay likelihood well to investigate whether they can effectively guide scheduling algorithms to robust shunting plans. The robustness measures are the *minimum total slack* (RM_3), the *minimum free slack* (RM_6)

Objective	**average**	**st. dev.**	$p = 0.05$	$p = 0.1$	$p = 0.2$
Feasibility	0.19	0.25	6.4	2.2	1.8
Feasibility + RM_3	0.06	0.14	3.1	1.4	1.2
Feasibility + RM_6	0.08	0.17	4.2	1.5	1.3
Feasibility + RM_9	0.02	0.04	2.5	1.5	1.2

Table 4.3: The probability of delays of the solutions per objective function for the *Kleine Binckhorst* instance.

and the *normal approximation* (RM_9).

Since the goal of the local search is to minimize the objective function, we add the value computed by a robustness measure multiplied by a weight $w_{RM} < 0$ to the objective. The negative weight ensures that the local search prefers higher robustness measure values. We have chosen the robustness measure weight sufficiently small to prefer conflict resolution over robustness, as the main objective of the local search is to find conflict-free solutions.

We have tested the performance of the local search with the different robustness objectives on two real-world instances of the shunting problem. The first instance consists of 19 trains arriving at the *"Kleine Binckhorst"* location. We simulate disturbances in this deterministic instance in the Monte Carlo simulation by sampling the durations of movement and service tasks from a log-normal distribution with the nominal duration of the tasks as the mean, and a standard deviation of 10% of the mean. The standard deviation is provided to the normal approximation robustness measure RM_9.

In the second instance, we have 24 trains that need to be parked and serviced on the larger shunting yard *"Grote Binckhorst"*. The disruptions in this instance are sampled from log-normal distributions for the duration of the movement and service activities with a standard deviation of 20% of the mean duration.

To study the effect of the robustness measures, we generated solutions using our local search by minimizing either only the conflict and movement penalties (*Feasibility*), or the base objective extended with one of the three robustness measures, resulting in 100 feasible shunting plans for each of the four configurations. The robustness of each of the plans — the probability of a delayed train departure — was estimated by sampling 10000 plan realizations in a Monte Carlo simulation.

Tables 4.3 and 4.4 summarize the results of the simulations. The columns **average** and **standard deviation** show statistics on the probability of delay in the solutions generated per objective function. The last three columns show the expected number of runs of the local search needed until a feasible solution with a delay probability less than p is found.

The results for both instances indicate that the probability of finding a robust solution can be improved significantly by adding any of the three robustness measures to the objective function. Although the normal estimation method takes slightly more time to evaluate than the two slack-based approaches, resulting in fewer iterations of the local search per time unit, the final solutions obtained by

Objective	**average**	**st. dev.**	$p = 0.05$	$p = 0.1$	$p = 0.2$
Feasibility	0.49	0.22	50.0	16.7	6.7
Feasibility + RM_3	0.26	0.16	33.3	7.1	2.5
Feasibility + RM_6	0.26	0.18	50.0	7.7	2.4
Feasibility + RM_9	0.15	0.12	20.0	3.1	1.2

Table 4.4: The probability of delays of the solutions per objective function for the *Grote Binckhorst* instance.

including the normal estimation into the objective are far more robust than those produced with the total and free slack robustness measures. In particular for the *Grote Binckhorst* instance, Table 4.4 shows that, on average, we will find a shunting plan with a delay probability of at most 10% after three runs of the local search algorithm with the normal estimation method in the objective, whereas more than seven runs are required with any of the other three objective functions.

4.4 Conclusion

In this chapter we have evaluated the performance of the robustness measures introduced in Chapter 3 in the setting of train unit shunting at railway yards. We compared the robustness measures with the results of a simulation study on shunting plans for two real-world shunting problems of NS. The experiments showed that approximating the makespan of the shunting plans with normal distributions is the strongest predictor of the robustness of shunting plans. The minimum slack robustness measures are good alternatives to the normal approximation approach when data of the variance of the uncertainty is unavailable.

Furthermore, we have compared several search objectives that incorporate these robustness measures with a basic, non-robust objective in a simulation of solutions generated with the local search described in Chapter 2. We have shown that the addition of a robustness measure based on estimating the completion times significantly improves the robustness of the solutions generated by the local search, outperforming minimum slack robustness measures.

A major topic for further research is extending the uncertainty in the scheduling problem to different types of disturbances. Examples of disturbances are train arrival sequences that deviate from the timetable or trains arriving in unexpected compositions. These types of disturbances occur frequently and often cannot be mitigated by delay propagation. Therefore, rescheduling algorithms have to be developed to cope with disruptions that cannot be absorbed by a shunting plan. We have started research on this topic in Onomiwo et al. (2020), where we propose local search methods that resolve conflicts in a shunting plan caused by disruptions without deviating far from the original schedule. The proposed algorithms often converge to feasible solutions within seconds, which is crucial in a real-world setting where rescheduling decisions have to be made as soon as possible. The schedules of the staff members in the modified shunting plans are highly similar to those in the origin solutions, thus increasing the acceptance of the rescheduling method by the personnel at the railway yards.

Chapter 5

Extending the Train Unit Shunting Problem with Staff Assignment

5.1 Introduction

In Chapter 2 we use a local search approach to generate solutions for the Train Unit Shunting Problem with Service Scheduling. In these solutions we do not explicitly roster the staff needed for the train activities. In particular, we assume that the number of *train drivers* or *train engineers* available on the shunting yard is sufficient to perform all train movements in the shunting plan. However, in practice personnel is a scarce resource, and hence their availability has a large impact on the feasibility of a shunting plan. Therefore, in this chapter we consider the integration of the staff scheduling into our yard planning approach.

The staff at the shunting yard, which consists of train drivers, mechanics and cleaning crews, typically works in eight-hour shifts. The availability of personnel within a single shift is constant, but the number of staff varies over the shifts. Additionally, half-way into their shift the personnel is entitled to a half-hour break.

Due to the sheer size of a shunting yard, the number of tasks that an employee can perform is severely limited by the walking distance between the locations of consecutive tasks. For example, if a driver is assigned to two train movements, and the destination of the first movement is far away from the start of the second movement, then the walking time between these two locations can easily be more than the total driving time of the two train movements combined. Even in the case that a train has two consecutive movements in opposite directions, and the driver continues to operate the same train, then the driver still has to walk from the driver's compartment at one end of the train to the other to have a clear view in the driving direction. As a result, reversing the direction of movement of a train with a single train driver can take up to 15 minutes for long train compositions.

As service and movement tasks take place at many locations on the railway yard, walking is often unavoidable. With some railway yard layouts the staff members, in particular the train drivers, might spend the majority of their shifts on walking between tasks. On such a yard a good staff assignment, in which the distance between tasks performed consecutively by the same staff member is small, is instrumental in performing all service and movement tasks in the shunting plan in a timely manner.

In practice the planners first construct a shunting plan and then try to find a feasible staff assignment. However, the staff assignment problem for the constructed shunting plan might be infeasible, in which case the shunting plan must be revised. In this chapter we focus on integrating the staff assignment problem with the shunting problem discussed in Chapter 2. By constructing a shunting plan in tandem with its staff assignment we can take the staff availability and walking durations into account when ordering the service and movement tasks to reduce the total walking time.

In this chapter we first give a formal introduction to the staff assignment problem in Section 5.2. We then propose two solution methods for the staff assignment, a list scheduling procedure and a decomposition approach, in Section 5.3 and discuss the embedding of the two staff assignment algorithms in the local search method presented in Chapter 2. We compare the two methods in Section 5.4 in an experimental study. We give some concluding remarks in Section 5.5.

5.2 Problem Description

Recall that the shunting problem at railway yards consists of six components:

1. matching incoming train units to outgoing train units;
2. (de-)coupling trains to form the correct train compositions for departure;
3. scheduling all required service activities such that they are completed before the trains depart;
4. parking the trains on the yard;
5. finding conflict-free paths for all train movements;
6. assigning staff to all the train activities.

In Chapter 2 we have discussed the first five problem components. A solution to these problem components — the *shunting plan* — consists of a set of activities A, a *partial order schedule* $\mathcal{POS}$ imposing precedence constraints on A, and the start time of every activity. As we did not consider the staff scheduling sub-problem in the previous chapter, we computed the start times of the activities based solely on the $\mathcal{POS}$.

However, in this chapter we assume that the activities in A require one or more skilled staff members. Since the staff assignment might impose additional precedence constraints on the activities, we now consider the problem of assigning both staff members and a start time to each of the activities, given the partial order schedule obtained from solving the first five shunting sub-problems.

The staff members are grouped by their skill set, i.e., the activity types that they are qualified to perform. Each activity $a \in A$ has

- a train t_a,
- a duration d_a,
- a track τ_a^{init} on which the activity is initiated,
- a track τ_a^{final} where the activity ends,
- a staffing requirement of r_a^T staff members of type T.

Note that $\tau_a^{init} \neq \tau_a^{final}$ if and only if the activity is a train movement. We have to assign to each activity a a sufficient number of staff members of the required type, as well as a feasible start time st_a and completion time $ct_a = st_a + d_a$.

For all available staff members in the planning horizon, we are given

- the start and end of their shift,
- the duration of their mandatory break,
- the interval in which the break has to take place, and
- their skill set type $T \in \mathcal{T}$. The skill sets are typically disjoint sets, i.e., train drivers may only move trains, cleaners clean the trains and mechanics are limited to repairing and inspecting the trains.

Furthermore, for each pair (τ_i, τ_j) of tracks we have the *walking duration* ω_{τ_i,τ_j}, which is the time required by a staff member to walk from track τ_i to track τ_j. Note that a more detailed model of the walking durations can be used if the exact locations of the train activities on the tracks (e.g. in meters) are known.

The staff assignment sub-problem is now to assign staff members and a start time to each of the activities in the shunting plan such that

- sufficient staff members with the correct skills are assigned to each activity;
- all activities of a staff member are scheduled within his or her shift and do not overlap;
- the start times of the activities satisfy the precedence constraints in the $\mathcal{POS}$;
- the time between activities in the schedule of a staff member is at least equal to the necessary walking duration;
- the break of each staff member is scheduled within the break interval, without overlap with other activities and without preemption.

For train drivers there are additional constraints associated with the scheduling of train movements. If a train driver is assigned to two consecutive movements of a train t in opposite directions, then the driver has to walk to the driver's compartment at the other end of the train to reverse the movement direction of the train. This *reversal* occurs after the completion of the first movement and before the start of the second movement. The duration of the reversal, d^t_{reversal}, depends linearly on the length of the train t.

Although a train movement only requires a single train driver, additional train drivers are allowed to travel along. Train drivers can travel with the train movements performed by other drivers to reach the locations of their next tasks faster. For example, suppose that the next task of a train driver is far from his current location. Instead of walking several kilometers, the driver might travel with another train movement that starts on a track near the driver's current location to get closer to his next task. Making a detour or stopping for (dis)embarking is not allowed during a train movement.

The objective of the train unit shunting problem — and therefore the staff scheduling sub-problem — is primarily to find a feasible shunting plan. However, the objective can be extended to include robustness metrics and staff preferences.

5.3 Solution Methods

To find shunting and service plans that satisfy the additional staff scheduling constraints introduced in the previous section, we propose two methods that can be integrated as sub-routines in the existing local search framework. These methods use the information in the partial order schedules generated by the local search to assign the train activities to their required resources, which includes the personnel.

The first method is a *list scheduling policy* that extends the movement scheduling heuristic in the local search described in Chapter 2. In this approach we maintain a sequence (or ordered list) of activities, which defines the order in which we evaluate the activities during the staff and start time assignment. For each activity in order, we compute the earliest start time based on the constraints in the partial order schedule, and the earliest time that a sufficient number of staff members is available to perform the activity given the starting times of the activities that are assigned earlier in the evaluation sequence.

In the second method, we *decompose* the problem into the sub-problems of assigning activities to the staff, and assigning start times to the activities. First, the staff assignment sub-problem, modeled as a maximum weighted matching problem, is solved. The solution to this sub-problem provides additional time lag constraints between the activities. These are then combined with the precedence constraints in the partial order schedule to compute the start times of the activities. If the start times assigned in the second step result in delayed train departures, we update the weights in the staff assignment problem and solve the two sub-problems again. This process is repeated, without modifying the solution to the first five problem components, until we find a good solution.

In the remainder of this section we will describe the two proposed methods in more detail.

5.3.1 List Scheduling Policy

In the list scheduling policy we use an ordered list L of the activities in the shunting plan. L is a linearization of the partial order schedule, i.e., a total ordering of the activities. The list L defines the order in which we will evaluate the activities to assign start times and staff members to them.

When we evaluate activity a_i in $L = (a_1, \ldots, a_n)$, activities a_j with $j < i$ have already been assigned a start time and staff member by the procedure. To compute the start time of a_i, we have to determine the availability of

- the train t_{a_i};
- infrastructure: cleaning platforms, train movement tracks, and switches;
- staff: train drivers, mechanics, and cleaners.

In the list scheduling approach we consider all the above as resources on which the activity has to be scheduled.

For each resource R required by a_i, we compute the set of feasible time windows T_R in which a_i can start on R given the duration of a_i and the activities a_j with $j < i$ that are already scheduled on R. For resources with a capacity larger than one, e.g. multiple train drivers or cleaning crews, the set of feasible start time windows consists of all time windows in which there is sufficient capacity available of the resource to process the activity.

The construction procedure of the set of feasible time windows depends on the type of resource. For the train of a_i, the feasible set consists only of a single time interval that starts after the maximum completion time of all predecessor activities of the train. For personnel required for a_i, we have to insert the activity in the schedule of a staff member such that sufficient time between activities is reserved for walking. The feasible time windows for the infrastructure are discussed in Section 2.3 in the time assignment of movement activities.

Once we have the set of feasible start times of each resource, we compute the start time of a_i as the earliest time that is feasible for all resources. Note that list L only indicates the priority of the activities, and not their actual order. Therefore, a_i can start before activity a_j with $j < i$ if the required resources and the partial order schedule allow this. We assign staff to activity a_i by retracing the staff members that contributed to the feasible time window at the computed start time of a_i.

Recall that the walking time of a train driver can be decreased by riding along with another train movement. To add this feature to the list scheduling policy, we take the scheduled train movements into account when computing the minimum time lag between two train movements assigned to the same driver.

Let a_i be the movement activity that we are currently scheduling in our list scheduling policy, and let train driver δ be available on time t_δ at track τ_δ. Then the train driver can

- either walk directly from track τ_δ to the initial track τ_i^{init} of a_i, which takes $\omega_{\tau_i^{init},\tau_\delta}$ time;
- or use one or more of the scheduled train movements to get to track τ_i^{init}.

For the latter case we evaluate all train movements that driver δ could use to take to reach the starting location of train movement a_i.

Let $(a'_1, \dots, a'_k)$ be the sequence of all train movements, ordered by starting time, that start after time t_δ and are scheduled prior to train movement a_i by the list scheduling policy. Note that the last condition is equivalent to $(a'_1, \dots, a'_k) \in \{a_1, \dots, a_{i-1}\}$.

We determine the earliest time that train driver δ can start train movement a_i using a dynamic programming approach. We denote by D^j_τ the earliest time that the driver reaches track τ with the possibility of traveling with the train movements from $(a'_1, \dots, a'_j)$, with $j \in \{1, \dots, k\}$. Additionally, we define $D^0_\tau = \omega_{\tau_\delta,\tau}$, which is the walking take from track τ_δ to track τ.

Then we can compute D^j_τ for all $j \in \{1, \dots, k\}$ and all tracks τ as

$$D^j_\tau = \begin{cases} D^{j-1}_\tau & \text{if } D^{j-1}_{\tau_j^{init}} > st_j \\ \min\left\{ct_j + \omega_{\tau_j^{final},\tau}, D^{j-1}_\tau\right\} & \text{otherwise.} \end{cases} \tag{5.1}$$

In the first case, train movement a'_j starts before the driver can reach the initial track τ_j^{init} of a'_j, hence the driver cannot use movement a'_j to get faster to track τ. In the latter case, the driver has the possibility to board train movement a'_j and walk from the destination track τ_j^{final} of a'_j to track τ.

The earliest time that driver δ can start movement activity a_i is then $D^k_{\tau_{a_i}^{init}}$, i.e., the earliest time that the driver can reach the initial track of a_i via any feasible combination of train movements.

5.3.2 Decomposition Heuristic

With the decomposition heuristic, we first solve the problem of assigning activities to personnel. The assignment gives us the schedules of the individual staff members, which we will combine with the precedence constraints from the partial order schedule to compute the start times of the activities in the second step.

We model the staff assignment problem for each type of personnel T (train drivers, mechanics and cleaners) separately as a maximum weighted matching problem in a bipartite graph. Let $A_T = \{a_1, \dots, a_n\}$ be the set of activities that require staff members of type T, and define k_T as the number of staff members available.

We construct the bipartite graph $G = (U, V, E)$ by adding for each activity a_i in A the vertices u_i to U and v_i to V. Additionally, U and V both have has k_T additional dummy vertices. A matching of the vertices in U to the vertices in V will model the predecessor-successor relations of the activities in the staff schedules. That is, if u_i is matched to v_j, then activity a_i directly precedes activity a_j in a staff schedule. The dummy vertices in U and V correspond to dummy activities representing the start and end of each schedule, respectively.

For every $u \in U$ and $v \in V$ we add an arc (u, v) if either:

- u is a dummy vertex and v is an activity vertex;
- u is an activity vertex and v is a dummy vertex;
- both u and v are activity vertices and activity a_u can precede activity a_v according to the partial order schedule.

For a matching to be feasible with respect to the whole shunting plan, the additional constraints should neither create cyclic precedence constraints on the activities nor cause delayed train departures.

The problem of cycles in the maximum matching arises from the partial order schedule, where we can have *anti-chains*, which are sets of activities that are incomparable, i.e., without a well-defined ordering in the $\mathcal{POS}$. These anti-chains can result in cycles in our matching problem. For example, suppose that we have two activities, a_i and $a_{i'}$, that are incomparable. Then the bipartite graph would contain the arcs $(u_i, v_{i'})$ and $(u_{i'}, v_i)$. If both arcs are selected in the matching, then a_i and $a_{i'}$ would have cyclic precedence constraints, which is clearly infeasible. To mitigate this problem we order the anti-chains during the construction of the graph G using the priority list L introduced in the previous sub-section. That is, we only include arcs from u_i to $v_{i'}$ if a_i has a higher priority than $a_{i'}$ in L. Note that the priorities only indicate the order of activities assigned *to the same staff member*. If two activities in an anti-chain are assigned to different staff members, then the activities can be performed simultaneously.

To find a maximum matching that is likely to satisfy the departure time constraints, we assign weights to the arcs in the bipartite graph. For each activity $a_i \in A_T$, we compute its earliest completion time ect_i and latest start time lst_i in the original partial order schedule. Then, for every arc (u, v) where u corresponds to an activity a_i and v to activity a_j, we set the weight of the arc to

$$w_{i,j} = lst_j - ect_i - mtl_{i,j}, \tag{5.2}$$

where $mtl_{i,j}$ is the minimum time lag due to walking between a_i and a_j. The weights of arcs involving dummy vertices are set to zero. With these weights, matchings in which the staff has sufficient time between their scheduled activities are preferred. Note that we cannot guarantee that a maximum weighted matching causes no departure delays, as the arc weights are only pair-wise indications of the feasibility of performing two activities consecutively. The weights cannot fully express whether sequences of more than two activities are feasible.

From the solutions of the staff assignment problems we obtain a set of minimum time lag constraints C between the activities that extend the precedence relations in the partial order schedule $\mathcal{POS}$. With these constraints we can assign the earliest start and completion times of the activities efficiently using a greedy approach. The earliest start time of an activity is the maximum of the completion time plus the minimum time lag over its direct predecessors in $C \cup \mathcal{POS}$. The order in which we assign the times to the activities follows from the priority list L. As L is a valid topological ordering of the activities with respect to the precedence constraints in C and $\mathcal{POS}$, this assignment order ensures that the start and completion time of an activity is assigned before evaluating any of its successors.

During the start times assignment we use the dynamic programming approach from Section 5.3.1, which allows drivers to travel with other train movements to reduce walking, to compute the minimum time between two consecutive movements assigned to the same train drivers.

After solving the staff and time assignment sub-problems, the solution might contain delayed train departure conflicts. To reduce the delays and improve our staff assignment, we will update the weights of the arcs, and solve the two sub-problems again. We select the matching variables that have their corresponding precedence relation on a critical path causing the delays, and reduce their weight by the total amount of departure delay resulting from the critical path. Then, we resolve the maximum weighted matching problem and again determine earliest start times of the activities. This procedure is repeated until we either find a feasible solution, fail to generate a new solution, or reach the predefined maximum number of iterations, i_{DH}. The best candidate solution found in the iterations is then returned by the decomposition heuristic.

5.3.3 Embedding in the Local Search

The two staff assignment heuristics are intended to extend a partial solution of the first five problem components defined in Section 5.2 — generated by the local search — to a full shunting plan including staff schedules. In addition to the $\mathcal{POS}$, the input of the two methods consists of a priority list L of the activities as well. Since the order of activities in the priority list affects the solutions produces by the two staff assignment algorithms, we have to allow the local search to modify the priority of the activities.

As described in Chapter 2, the local search has several neighborhoods that change the order of the activities in the partial order schedule. When the local search modifies the ordering in the $\mathcal{POS}$, we update the priorities of the activities such that the priority list remains a linearization of the partial ordering. Additionally, we introduce two new neighborhoods that change the ordering of L without affecting the $\mathcal{POS}$ by **shifting** and **swapping** activities, respectively.

5.4 Experimental Setup and Results

To compare the two solution methods, we will evaluate their performance on a set of instances generated for a real-world shunting yard. The *"Kleine Binkchorst (KBH)"*, shown in Figure 2.1, is a shunting yard of the NS near the central station of The Hague. For this location, we have generated 300 instances in which 13 to 15 train units arrive during the evening and have to depart the next morning. The

Method	Solved Instances	Average Computation Time (s)
No Drivers	300	32
Complete *LSP*	276	54
Complete *DH*	196	414
Partial *LSP*	286	116
Partial *DH*	195	245

Table 5.1: The results for 300 instances of the *Kleine Binckhorst* yard in the Netherlands. The average computation time of feasible solutions is listed in the table.

arrival and departure times are sampled from an empirical distribution based on data provided by NS. All trains have to be cleaned internally.

For the case without drivers, feasible solutions for 300 instances are found by the local search algorithm described in Chapter 2, as is shown in Table 5.1. To model the staff assignment problem, we included three train drivers that will be available during the entire planning horizon. The walking distances are provided by NS based on real-world data and range from 2 to 20 minutes, depending on the physical location of the tracks.

We tested four configurations of the local search on these instances extended with the staff requirements. In the first two configurations we call the staff assignment sub-routine (either the list scheduling policy *LSP* or the decomposition heuristic *DH*) for every solution that the local search explores. We will refer to these two configurations as the *complete LSP* and the *complete DH* approaches. In the two other configurations, we first search for a feasible solution without any staff assignment, similar to the baseline case where we did not include the train drivers in the instances. Once a feasible solution without personnel is found, we run either one of the staff assignment heuristics. If the resulting solution contains delays, we continue the local search algorithm with the staff assignment sub-routine. These two configurations are listed in Table 5.1 as the *partial LSP* and *partial DH* methods. On each instance we ran the four local search variants until a feasible solution was found, or the maximum computation time of 1800 seconds was reached.

Table 5.1 shows the number of solved instances as well as the computational results of our experiments. All instances with train drivers have been solved successfully by at least one of two list scheduling configurations of the local search. The list scheduling policy outperforms the decomposition heuristic for both the complete and partial local search variants. The decomposition heuristic fails to find feasible solutions for one-third of the instances, whereas the list scheduling algorithm solves most of the instances relatively fast. This might indicate that the decomposition approach is not able to update the edge weights properly to converge to a good schedule for the train drivers. The longer computation time of the decomposition approach is most likely due to constructing and solving the matching problem every iteration.

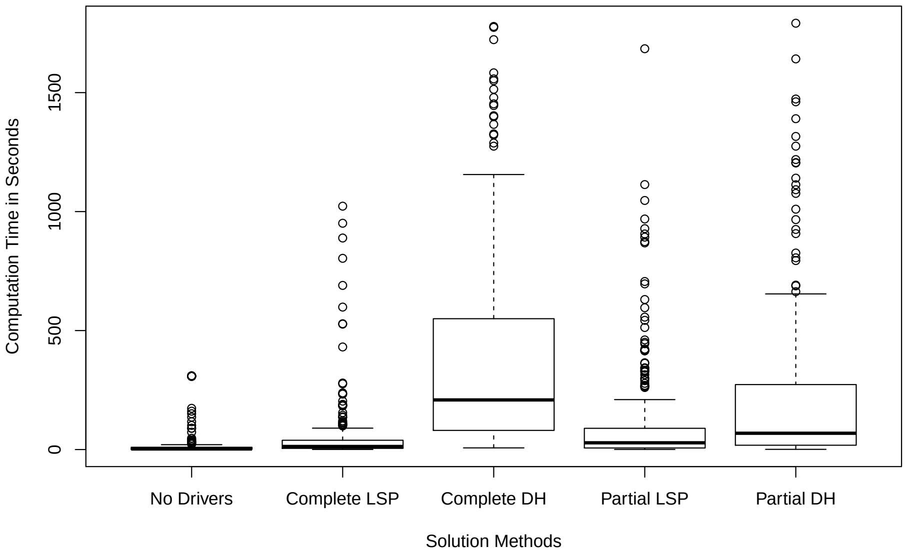


Figure 5.1: The computation time of solved instances of the five local search configurations listed in Table 5.1.

The differences between starting directly with the staff assignment or first searching for a feasible solution without staff are much smaller. In the case of the LSP approach, starting from a feasible solution without staff increases the number of instances that can be solved at the cost of doubling the computation time. This suggests that adapting the solution without staff to the staffing constraints might require either many small modifications, or several high-cost intermediate solutions that are unattractive for the local search to explore.

Of the four local search configurations, the variant in which we schedule the staff with the list scheduling policy for every candidate solution shows the best performance. The computation time of this configuration is close to the local search without staff assignment, and 92% of the instances are solved.

5.5 Conclusion

In this chapter, we have studied the extension of the train unit shunting problem with staffing constraints. We proposed two methods that construct staff schedules for a given partial ordering of the activities on the shunting yard. The first method implements a list scheduling policy to distribute the activities of the available personnel, whereas the second approach decomposes the problem into a staff assignment and a time assignment sub-problem that are solved iteratively. These methods can be used in conjunction with the local search presented in Chapter 2 to find feasible shunting plans that fully integrate all components of the planning problem at the railway yards.

We studied the performance of the solution methods on a set of 300 realistic instances of the "Kleine Binckhorst" shunting yard. The experiments show that the list scheduling approach outperforms the decomposition heuristic, and that the former solves 92% of the instances in reasonable time.

Chapter 6

Exact Methods for Train Driver Assignment at Railway Yards

6.1 Introduction

In Chapter 5 we introduced the staff assignment problem at railway yards. We proposed two heuristic methods that try to assign staff members and start times to activities, which are constrained by a partial order schedule, such that the temporal constraints and staff requirements are satisfied. The heuristics can be combined with the local search approach described in Chapter 2, which constructs the partial order schedules of the activities, to find feasible shunting plans with staff assignments.

In practice the scheduling of the train drivers is the main bottleneck of the staff assignment problem. Cleaning and maintenance tasks are often limited to a few locations on the railway yard due to the requirement of specialized facilities or tools, whereas train movements take place all over the yard. Therefore, the efficiency of the schedules of the train drivers is a major concern of the planners.

To evaluate the quality of the solutions produced by the heuristics, we can compare them to optimal train driver assignments for the partial order schedule. In this chapter we propose exact solution methods based on mixed integer linear programming for the train driver assignment problem and several extensions of this problem. The problem extensions consist of robust formulations of the basic problem, allowing drivers to join scheduled train movements as passengers, and limiting the availability of the train drivers to predetermined shifts.

This chapter is organized as follows. We start with an overview of the train driver assignment problem in Section 6.2, and formulate MIP models of two variants of the problem in Sections 6.3 and 6.4. We compare solutions constructed by the heuristics in Chapter 5 with the optimal train driver assignment of the exact models in Section 6.5. We conclude this chapter with recommendations for future research in Section 6.6.

6.2 Problem Description

Similar to the general staff assignment problem at railway yards, described in Section 5.2, the input of the train driver assignment problem is a set of train drivers who are available to operate the trains, and a shunting plan that contains both the train movements that have to be performed and a set of minimum time lags between the movements. The minimum time lags arise from the service activities scheduled in the shunting plan as well as the train safety regulations, which specify a minimum time between train movements over the same track. Each train movement has to be operated by one train driver.

Let A be the set of train movements in the shunting plan, consisting of movements $a_1, \ldots, a_n$. From the partial order schedule of all activities in the shunting plan we can derive the set of minimum time lag constraints between the train movements. We denote this set by C, where $mtl_{ij} \in C$ is the minimum time lag between the completion times of a_i and a_j. Note that there is a time lag between a_i and a_j if and only if a_j is a direct successor of a_i in the partial order schedule. Furthermore, for each train movement a_i we can derive its earliest completion time ect_i and latest completion time lct_i in the shunting plan from the arrival and departure times of the trains, and the minimum time lags in C.

To operate the trains there are m train drivers present during the planning horizon at the railway yard. Each driver $k \in \{1, \ldots, m\}$ is available during his or her shift, which starts at s_k and ends at e_k. The drivers all start and end their shifts at the same location on the railway yard. The walking time from the destination of movement a_i to the origin of movement a_j is denoted as ω_{ij}. We further use the following notation:

$$
\begin{aligned}
A &= \text{set of train movements } \{a_1, \ldots, a_n\} \text{ that have to be operated,} \\
dur_i &= \text{duration of movement } a_i, \\
ect_i &= \text{earliest completion time of movement } a_i, \\
lct_i &= \text{latest completion time of movement } a_i, \\
C &= \text{set of minimum time lag constraints between movement pairs, with} \\
mtl_{ij} \in C &= \text{minimum time lag from completion time of } a_i \text{ to completion time of } a_j, \\
\omega_{ij} &= \text{minimum walking time from } a_i \text{ to } a_j, \\
\omega_{0i} &= \text{minimum walking time from shift start location to } a_i, \\
\omega_{i0} &= \text{minimum walking time from } a_i \text{ to shift end location.}
\end{aligned}
$$

The goal of the train driver assignment problem is to assign train drivers and start times to the train movements in A such that each movement is operated by a train driver, and the start times satisfy the walking and time lag constraints. Although feasibility is the main objective, planners try to construct train driver assignments that use as few drivers as possible. The surplus of train drivers may be kept in reserve during operations to quickly respond to unexpected events such as the delayed or unplanned arrival of trains on the railway yard.

6.3 Continuous Driver Shift Models

We first study variants of the train driver assignment problem in which the shifts of the train drivers all span the entire planning horizon. That is, we assume that all shifts start before the first train arrives and end after the last departure.

In the models we will decide on the completion time $c_i \geq 0$ of each train movement $a_i \in A$. Furthermore, we have to assign the movements to the train drivers. We model the assignment as a flow problem in which the flow through the train movements defines the staff schedules.

Let a_0 be a dummy movement representing the start and end of the shunting plan, and define $A^+ = A \bigcup \{a_0\}$. For each pair of movements $a_i, a_j \in A^+$, with $a_i \neq a_j$, we define the decision variable

$$x_{i,j} = \begin{cases} 1 & \text{if } a_i \text{ directly precedes } a_j \text{ in the schedule of a train driver,} \\ 0 & \text{otherwise.} \end{cases}$$

By assigning to each $a_i \in A$ a single predecessor and successor we construct the sequences of the train movements in the driver schedules. Each schedule starts and ends with a_0.

To formulate the staff assignment problem as a mixed integer linear program, we denote the following properties of the $\mathcal{POS}$ and the driver schedules as

$$mtl'_{ij} = \begin{cases} mtl_{ij} & \text{if } mtl_{ij} \in C \\ ect_j - lct_i & \text{otherwise,} \end{cases}$$

$$\omega'_{ij} = \max\{\omega_{ij} + dur_j - mtl'_{ij}, 0\}$$

The problem of finding a feasible driver assignment can then be formulated as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} x_{0,i} & & & (6.1)\\
\text{s.t.} \quad & c_j - c_i - \omega'_{ij} x_{i,j} \geq mtl'_{ij} & & \forall a_i, a_j \in A & (6.1a)\\
& \sum_{j=0}^{n} x_{i,j} = 1 & & \forall a_i \in A & (6.1b)\\
& \sum_{j=0}^{n} x_{j,i} = 1 & & \forall a_i \in A & (6.1c)\\
& c_i \in [ect_i, lct_i] & & &\\
& x_{i,j} \in \{0, 1\}. & & &
\end{aligned}$$

In Model 6.1 the minimum time lag between activities is enforced by (6.1a). The minimum necessary time lag ω'_{ij} is included if and only if a_i is the direct predecessor of a_j in some train driver schedule. Each movement $a_i \in A$ is included in a single staff schedule due to the in- and outflow constraints (6.1b) and (6.1c).

Note that in any feasible solution the precedence graph induced by C and the driver schedules is always acyclic due to the constraints in (6.1a).

6.3.1 Robustness

Since the railway yard is a very dynamic environment with many possible disturbances, we strive for resilient solutions. In Chapters 3 and 4 we studied several metrics that quantify the robustness of the shunting plans. Although robustness measures based on normal distribution approximations cannot easily be expressed in a linear model, some of the slack-based approaches can be formulated using linear constraints.

We will illustrate this with two robust formulations of Model 6.1. In both formulations we assume that the number of train drivers in the problems are fixed; we denote the number of drivers by m.

In our first model, we use the *free slack* times — the maximum amount of time an activity can be delayed without affecting other activities — as a measure for the robustness. We denote this by fs_i and try to maximize the sum in our objective function. In Model 6.2 we formulate the mixed integer linear program of the train driver assignment problem with free slack times as

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n} fs_i & & & (6.2)\\
\text{s.t.} \quad & c_j - c_i - fs_i - \omega'_{ij} x_{i,j} \geq mtl'_{ij} & & \forall a_i, a_j \in A & (6.2a)\\
& \sum_{j=0}^{n} x_{i,j} = 1 & & \forall a_i \in A & (6.2b)\\
& \sum_{j=0}^{n} x_{j,i} = 1 & & \forall a_i \in A & (6.2c)\\
& c_i + fs_i \leq lct_i & & \forall a_i \in A & (6.2d)\\
& \sum_{i=1}^{n} x_{0,i} \leq m & & & (6.2e)\\
& c_i \geq ect_i & & &\\
& x_{i,j} \in \{0,1\}. & & &
\end{aligned}$$

In this formulation the free slack of activity a_i is constrained by the minimum time lag relations in (6.2a) and the latest completion time of a_i in (6.2d). Constraint (6.2e) provides the upper bound on the number of train drivers that can be assigned to the activities.

In our second model we measure the robustness using the *total slack* times of the activities. In the context of a scheduling problem with deadlines the total slack of an activity represents the maximum amount of time that the activity can be delayed without violating any of the deadline constraints. The total slack can be expressed as the difference between the largest possible completion time and the smallest possible completion time of the activity in the solution. We denote the smallest completion time of activity a_i by c_i^- and the largest completion time by c_i^+. Note that c_i^- is the smallest completion time of a_i in a train driver assignment, whereas the input parameter ect_i is earliest completion time of a_i in the partial order schedule of the shunting plan without train drivers.

The objective function of the second model is the maximization of the sum of the total slack times of the activities. We formulate the train driver assignment with total-slack-based robustness as

$$
\begin{aligned}
\max \quad & \sum_{i=1}^{n} \left(c_i^+ - c_i^-\right) && && (6.3)\\
\text{s.t.} \quad & c_j^- - c_i^- - \omega'_{ij} x_{i,j} \geq mtl'_{ij} && \forall a_i, a_j \in A && (6.3a)\\
& c_j^+ - c_i^+ - \omega'_{ij} x_{i,j} \geq mtl'_{ij} && \forall a_i, a_j \in A && (6.3b)\\
& \sum_{j=0}^{n} x_{i,j} = 1 && \forall a_i \in A && (6.3c)\\
& \sum_{j=0}^{n} x_{j,i} = 1 && \forall a_i \in A && (6.3d)\\
& c_i^+ - c_i^- \geq 0 && \forall a_i \in A && (6.3e)\\
& \sum_{i=1}^{n} x_{0,i} \leq m && && (6.3f)\\
& c_i^- \geq ect_i && &&\\
& c_i^+ \leq lct_i && &&\\
& x_{i,j} \in \{0,1\}. && &&
\end{aligned}
$$

In Model 6.3 the constraints (6.3a) and (6.3b) ensure that the smallest and largest completion time variables satisfy the minimum time lag constraints imposed by the shunting plan and the train driver assignment. The total slack cannot be negative in a feasible solution, which is modeled by constraint (6.3e). Similar to constraint (6.2e) of Model 6.2 we bound the maximum number of drivers that can be assigned with constraint (6.3f).

When the shunting plan and the train driver assignment is implemented during operations, all activities will be performed as early as possible. If no disturbances occur, then each activity a_i will be completed at time c_i^-.

6.3.2 Allowing passengers

To reduce the distance that train drivers have to travel on foot, they can board the trains operated by their colleagues to move more efficiently over the railway yard, as described in Section 5.2. To allow multiple train drivers on the same train movement, we introduce, next to $x_{i,j}$, for each pair of movements $a_i, a_j \in A^+$ the decision variable

$$y_{i,j} = \text{ the number of driver schedules in which } a_i \text{ directly precedes } a_j,$$

with $y_{i,j} \geq 0$ and integral.

We formulate the driver assignment problem extended with passenger options as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} y_{0,i} && && (6.4)\\
\text{s.t.} \quad & c_j - c_i - w'_{ij} x_{i,j} \geq mtl'_{ij} && \forall a_i, a_j \in A && (6.4a)\\
& \sum_{j=0}^{n} y_{i,j} \geq 1 && \forall a_i \in A && (6.4b)\\
& \sum_{j=1}^{n} (y_{i,j} - y_{j,i}) = 0 && \forall a_i \in A && (6.4c)\\
& y_{i,j} - x_{i,j} \geq 0 && \forall a_i, a_j \in A && (6.4d)\\
& m x_{i,j} - y_{i,j} \geq 0 && \forall a_i, a_j \in A && (6.4e)\\
& c_i \in [ect_i, lct_i] && &&\\
& x_{i,j} \in \{0,1\} && &&\\
& y_{i,j} \in \mathbb{N}_0 && &&
\end{aligned}$$

Similar to Model 6.1, we have the minimum time lag constraints in (6.4a). The constraints in (6.4b) ensure that at least one train driver is assigned to each movement. Constraint (6.4c) ensures that the number of drivers embarking and disembarking for train movement a_i remains balanced. The two constraints (6.4d) and (6.4e) model the logical relation between $x_{i,j}$ and $y_{i,j}$: $y_{i,j} \geq 1 \Leftrightarrow x_{i,j} = 1$.

Note that the model does not decide which train driver operates the train when multiple drivers are present during a train movement. However, since we assume that each train driver has the skill set to operate all train types, the distinction between operator and passenger does not impact the staff schedules.

6.4 Non-identical Driver Shifts Models

In the previous section we assumed that each train driver is available during the complete planning horizon. However, in practice the train drivers work in shifts of six to nine hours. The number and start times of the train driver shifts are scheduled before the shunting process at the railway yard is planned, and thus is input to the train driver assignment problem. The shifts can overlap in time and the number of simultaneous shifts can vary during the planning horizon. We assume that all driver shifts start and end at the same location on the railway yard.

We denoted the set of train driver shifts by K, which consists of m shifts. Each shift $k \in K$ has a start time s_k and an end time e_k. Instead of a single dummy movement a_0 that models the start and end of all shifts, we introduce for each shift $k \in K$ a dummy movement a_{n+k} that represents the start and end of shift k.

Similar to the models proposed earlier in this chapter, we model the train driver assignment as a flow circulation problem in which the flow through the train movements defines the driver schedules. The $x_{i,j}$ decision variables model the flow, and the flow corresponding to the driver schedule of shift $k \in K$ starts and ends at activity a_{n+k}. However, since the shifts are not assumed to be identical, we have to ensure that the flow which starts in dummy activity a_{n+k} does not pass through any of the other dummy activities. For example, suppose that we have two shifts, which are scheduled for the time intervals $[9:00, 17:00]$ and $[12:00, 20:00]$. Then we have to prevent train driver assignments with flows from $9:00$ to $20:00$ or from $12:00$ to $17:00$, as these flows do not correspond to shifts.

To avoid solutions with invalid flows we introduce decision variables to explicitly assign shifts to activities. In addition to the $x_{i,j}$ variables, we have three binary variables for each activity-shift pair in our model to represent the assignment of drivers to activities:

$$z_{i,k} = \begin{cases} 1 & \text{if movement } a_i \text{ is operated by the driver of shift } k, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{n+k,i} = \begin{cases} 1 & \text{if the driver of shift } k \text{ starts with } a_i, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{i,n+k} = \begin{cases} 1 & \text{if the driver of shift } k \text{ ends with } a_i, \\ 0 & \text{otherwise.} \end{cases}$$

Furthermore, for each shift we add a binary decision variable

$$x_{n+k,n+k} = \begin{cases} 1 & \text{if the driver of shift } k \text{ is not assigned to any activity in } A, \\ 0 & \text{otherwise,} \end{cases}$$

to model that a shift is not used in the train driver assignment. Let $ect_{i,k}$ and $lct_{i,k}$ be the earliest and latest completion times of a_i if a_i is the first or last activity in shift k, respectively:

$$\begin{aligned} ect_{i,k} &= s_k + \omega_{0i} + dur_i, \\ lct_{i,k} &= e_k - \omega_{i0}. \end{aligned}$$

We formulate the train driver assignment problem with non-identical shifts in

Model 6.5,

$$\min \quad \sum_{k=1}^{m}\sum_{i=1}^{n} x_{n+k,i} \tag{6.5}$$

$$\text{s.t.} \quad c_j - c_i - \omega'_{ij} x_{i,j} \geq mtl'_{ij} \qquad \forall a_i, a_j \in A \tag{6.5a}$$

$$c_i - \sum_{k=1}^{m} (ect_{i,k} - ect_i)\, x_{n+k,i} \geq ect_i \qquad \forall a_i \in A \tag{6.5b}$$

$$c_i - \sum_{k=1}^{m} (lct_{i,k} - lct_i)\, x_{i,n+k} \leq lct_i \qquad \forall a_i \in A \tag{6.5c}$$

$$\sum_{j=1}^{n+m} x_{i,j} = 1 \qquad \forall a_i \in A \tag{6.5d}$$

$$\sum_{j=1}^{n+m} x_{j,i} = 1 \qquad \forall a_i \in A \tag{6.5e}$$

$$x_{n+k,n+k} + \sum_{i=1}^{n} x_{n+k,i} = 1 \qquad \forall k \in K \tag{6.5f}$$

$$x_{n+k,n+k} + \sum_{i=1}^{n} x_{i,n+k} = 1 \qquad \forall k \in K \tag{6.5g}$$

$$\sum_{k=1}^{m} z_{i,k} = 1 \qquad \forall a_i \in A \tag{6.5h}$$

$$z_{i,k} - z_{j,k} + x_{i,j} + x_{j,i} \leq 1 \qquad \forall a_i, a_j \in A, \forall k \in K \tag{6.5i}$$

$$z_{j,k} - z_{i,k} + x_{i,j} + x_{j,i} \leq 1 \qquad \forall a_i, a_j \in A, \forall k \in K \tag{6.5j}$$

$$c_i \in [ect_i, lct_i] \tag{6.5k}$$

$$x_{i,j} \in \{0,1\} \tag{6.5l}$$

$$z_{i,k} \in \{0,1\}. \tag{6.5m}$$

In this model we minimize the number of train drivers assigned to activities. Constraints (6.5a) model the minimum time lags between activities that are imposed by the staff assignment and the partial order schedule.

Since the shifts do not span the entire planning horizon, it is necessary to include in the model that the drivers start and end their shift at a fixed location on the railway yard. Constraints (6.5b) ensure that the drivers have enough time to walk from that location to start locations of their first activities. Similarly, the drivers can walk from the last activity in their shifts back to the central location within theirs shifts due to Constraints (6.5c).

Constraints (6.5d) and (6.5e) model that each activity $a_i \in A$ has a single successor and predecessor in the shift to which it is assigned. The constraint that each shift is either empty or contains a first activity is modeled by (6.5f). A similar constraint for the last activity in each shift is given by (6.5g).

Each activity is assigned to exactly one driver shift by constraint (6.5h). Constraints (6.5i) and (6.5j) ensure that if a_i directly precedes or succeeds a_j in a driver shift, then a_i and a_j are assigned to the same driver shift. That is,

$$x_{i,j} = 1 \vee x_{j,i} = 1 \Rightarrow \forall k \in K : z_{i,k} = z_{j,k},$$

which is equivalent to

$$\forall k \in K : |z_{i,k} - z_{j,k}| \leq 1 - x_{i,j} - x_{j,i}.$$

Model 6.5 gives a complete description of the train driver assignment problem with shifts. The constraints that each activity a_i must be completed within the shift to which it is assigned are modeled implicitly by the constraints (6.5a), (6.5b) and (6.5c). In Model 6.5' we extend Model 6.5 by explicitly constraining the completion times c_i of the activities to the start and end times of the shifts.

$$\begin{array}{llrr}
\min & \displaystyle\sum_{k=1}^{m}\sum_{i=1}^{n} x_{n+k,i} & & (6.5') \\
\text{s.t.} & c_j - c_i - \omega'_{ij} x_{i,j} \geq mtl'_{ij} & \forall a_i, a_j \in A & (6.5a) \\
 & \vdots & & \vdots \\
 & z_{i,k} \in \{0,1\}, & & (6.5m)
\end{array}$$

$$c_i - \sum_{k=1}^{m} (s_k - dur_i - ect_i)\, z_{i,k} \geq ect_i \qquad \forall a_i \in A \qquad (6.5n)$$

$$c_i - \sum_{k=1}^{m} (e_k - lct_i)\, z_{i,k} \leq lct_i \qquad \forall a_i \in A \qquad (6.5o)$$

The constraints (6.5n) and (6.5o) ensure that the activities are operated within the time intervals of the shifts. Although these constraints are redundant with respect to the integral solution space, the explicit restrictions of the completion times improve the LP-relaxation of the model. This can reduce the computation time needed to solve the model, as shown in Section 6.5.

6.4.1 Branch-and-price formulation

The MIP formulation proposed in Model 6.5 introduces a large number of constraints to schedule the activities in the available driver shifts. In particular, the model contains constraints (6.5i) and (6.5j) for every combination of activity pairs and shifts. Models with large sets of constraints can be challenging to solve efficiently. As an alternative to the flow-based approach used in the previous models, we can formulate the problem as a covering problem in which we assign complete *driver schedules* of activities to the driver shifts. This leads to a model with fewer constraints. The major disadvantage of this approach is in the number of decision variables; the number of possible activity schedules that can be assigned to the shifts increases exponentially with the number of activities.

To cope with the large number of decision variables we apply the branch-and-price method. With this (mixed) integer linear programming technique we start with a small subset of the decision variables, or columns, and solve the restricted version of the LP relaxation of the master problem. We then search for columns that might reduce the objective value by solving a sub-problem, which is known as the pricing problem, and add these to the master problem. We continue to solve the master and pricing problem alternately until we find no new potentially beneficial columns in the pricing problem. Then we solve the integral version of the master problem. This process is repeated for each node in the branch-and-bound tree.

In the master problem of the branch-and-price approach, we decide on the completion time $c_i \geq 0$ of each activity $a_i \in A$. Furthermore, we have to assign the activities to the train driver shifts. We model the schedules of individual train driver shifts as sequences of the activities in A. In such an individual driver schedule we only describe the sequence of activities to be done; the times at which these are done are decided by the master problem.

Let Π_k be the set of possible driver schedules for shift k, then we can construct a feasible driver assignment by selecting for each shift k a schedule $\pi_l \in \Pi_k$ such that all activities are covered and completed before their deadline. We represent the decision of selecting driver schedule $\pi_l \in \Pi_k$ by the binary decision variable $ds_{k,l}$, where

$$ds_{k,l} = \begin{cases} 1 & \text{if schedule } \pi_l \text{ is chosen for train driver shift } k, \\ 0 & \text{otherwise.} \end{cases}$$

To formulate the train driver assignment problem with shifts as a mixed integer linear program, we denote the following properties of the train driver schedules as

$$a_{i,l} = \begin{cases} 1 & \text{if } a_i \text{ is in schedule } \pi_l \\ 0 & \text{otherwise} \end{cases}$$

$$r_{ijl} = \begin{cases} 1 & \text{if } a_i \text{ directly precedes } a_j \text{ in schedule } \pi_l \\ 0 & \text{otherwise} \end{cases}$$

The problem of finding a feasible train driver assignment that minimizes the number of shifts assigned to activities is formulated in Model 6.6.

$$\min \quad \sum_{k=1}^{m} \sum_{l \in \Pi_k} ds_{k,l} \tag{6.6a}$$

$$\text{s.t.} \quad \sum_{k=1}^{m} \sum_{l \in \Pi_k} a_{i,l} ds_{k,l} = 1 \qquad \forall a_i \in A \qquad (\alpha_i) \tag{6.6b}$$

$$c_j - c_i - \omega'_{i,j} \sum_{k=1}^{m} \sum_{l \in \Pi_k} r_{i,j,l} ds_{k,l} \geq mtl'_{i,j} \qquad \forall a_i, a_j \in A \quad (\beta_{ij}) \tag{6.6c}$$

$$\sum_{l \in \Pi_k} ds_{k,l} \leq 1 \qquad \forall k \in K \qquad (\gamma_k) \tag{6.6d}$$

$$c_i \in [ect_i, lct_i] \tag{6.6e}$$

Constraints (6.6b) ensure that each activity is included in one of the driver shifts. The minimum time lag constraints imposed by the partial order schedule and the driver schedules are modeled by (6.6c). At most one sequence of activities is assigned to each driver shift due to constraints (6.6d).

The promising driver schedules π_l are identified in the pricing problem. The values of the dual variables of the constraints (6.6b), (6.6c) and (6.6d) are denoted by α_i, $\beta_{i,j}$ and γ_k, respectively. The reduced cost of a schedule π_l of driver shift $k \in K$ is given by

$$1 - \sum_i \alpha_i a_{i,l} + \sum_{i,j} \beta_{i,j} \omega'_{i,j} r_{i,j,l} - \gamma_k, \tag{6.7}$$

and π_l is added to the restricted master problem if its reduced cost is less than zero.

The pricing problem is then to construct an ordered subset of the activities for the driver shift that maximizes the weight

$$\sum_i \alpha_i a_{i,l} - \sum_{i,j} \beta_{i,j} \omega'_{i,j} r_{i,j,l}, \tag{6.8}$$

subject to the feasible completion time intervals $[ect_i, lct_i]$ of each activity a_i, the minimum time lag constraints between activities derived from the $\mathcal{POS}$, the walking time of the train driver, and the time interval of the shift.

Finding an optimal solution to this pricing problem is NP-hard. In Van den Akker et al. (2012) the authors propose a local search method to solve a similar pricing problem. They apply the column generation technique to a parallel machine scheduling problem with release dates, deadlines and generalized precedence constraints. The corresponding pricing problem is to find a feasible schedule for a single machine that minimizes the reduced cost such that all precedence and temporal constraints are satisfied.

An alternative approach is to first discretize the problem to a time-indexed formulation. We can then model the discretized pricing problem as a single-source path-finding problem in a time-expanded graph. In this directed graph we have two vertices representing the start and end of the shift. We denote these vertices by v_s and v_e, respectively. Furthermore, for every activity a_i and every time step $t \in [s_k, e_k]$ we add a vertex $v_{i,t}$ to the graph that represents that activity a_i is completed at time t.

The arcs in the graph model the sequences of activities that can be operated by the driver. We add an arc from $v_{i,t}$ to $v_{j,t'}$ if the driver can complete a_i at time t followed by a_j at time t' without violating any of the constraints:

- $t \in [ect_i, lct_i]$;
- $t' \in [ect_j, lct_j]$;
- the driver can walk from a_i to a_j and operate a_j in $t' - t$ time;
- completing a_i at time t and a_j at time t' satisfies the minimum time lag constraints imposed by the $\mathcal{POS}$.

The weight of an arc from activity a_i to a_j is set to $\alpha_j - \beta_{i,j}\omega'_{i,j}$.

In addition to the arcs between activities we also connect the two shift vertices to the activities. An arc from the start-of-shift vertex v_s to vertex $v_{i,t}$ is added to the graph if assigning a_i as the first activity in the shift and completing a_i at time t is feasible. The weight of this arc is α_i. Similarly, we add an arc with weight 0 from vertex $v_{i,t}$ to vertex v_e if the driver can finish the shift by completing a_i at time t and returning to the end-of-shift location.

The resulting time-expanded graph is an acyclic directed graph. The objective of the path-finding problem in this graph is to construct a path from v_s to v_e that maximizes the total weight of the path.

Note that the driver schedules constructed in the pricing problem of our proposed branch-and-price method consist of sequences of activities; the completion times of the activities are decision variables in the master problem. In the pricing problem we only check the completion times to ensure that each driver schedule π_l satisfies the temporal constraints.

An alternative formulation is to assign the completion times of the activities in the driver schedules. In this formulation, the completion times of activities are decision variables in the pricing problem instead of the master problem. Although this model has no completion time decision variables in the master problem, the set of possible driver schedules is much larger. Furthermore, the completion times that are assigned in the pricing problem have to be compatible with respect to the temporal constraints imposed by the $\mathcal{POS}$ in the master problem. We have tested both the proposed formulation with activity sequences and the option of assigning the completion times in the driver schedules on small instances, but the latter turned out to be significantly slower, and therefore we did not pursue this line of research.

6.5 Experimental Setup and Results

To evaluate the quality of the staff assignments produced by the list scheduling heuristic proposed in Chapter 5, we compare the train driver assignments of these

Instance	Train Movements	Driver Shifts	Max. Simultaneous Shifts
Alkmaar	73	11	4
Rotterdam	126	31	12
Eindhoven	276	18	8

Table 6.1: Characteristics of the problem instances of three major stations in the Netherlands.

solutions to the optimal assignments computed with the MIP models formulated in the previous section. For three large real-world instances we have constructed shunting plans and staff assignments using the local search approach combined with the list scheduling policy, which was denoted in Chapter 5 by Complete *LSP*. Each of the instances is a standard week day on a major station in the Netherlands. The number of train movements that have to be operated by a train driver and the number of driver shifts in the instances are listed in Table 6.1. The last column of the table shows the maximum number of train drivers that are simultaneously on the railway yard during the planning horizon.

By design the list scheduling policy will greedily assign whomever is available first to the train movements. This often results in train driver assignments in which all drivers are assigned, but some drivers having an almost empty schedule. To determine the minimum number of train driver shifts required by the list scheduling policy to operate all train movements, we repeatedly solve the problem instance with the policy with decreasing number of available driver shifts. That is, we solve the problem instance, remove the train driver shift with the largest amount of idle time in the current solution from the instance, and try to solve this restricted problem instance. We repeat this process until the list scheduling policy fails to find a feasible train driver assignment. The computation times of the list scheduling heuristic presented in the tables are time measurements over the complete iterative process.

6.5.1 Continuous driver shifts

We first compare the heuristic with Models 6.1 and 6.4, which model the train driver assignment problem with identical shifts that span the full planning horizon. We compare solutions of the heuristic approach and the two exact methods on the number of train drivers required to perform all train movements as well as the total walking time of the drivers in the solutions. The MIP models of the problem instances are solved using Gurobi 9.1.

Table 6.2 shows that the number of drivers needed to operate the trains is in all cases significantly lower in the optimal solutions when compared to the assignments produced by the heuristic. The list scheduling policy requires all available train drivers in the *Alkmaar* and *Eindhoven* instances to construct a feasible train driver assignment. The passengers options added in Model 6.4 do not lead to lower driver requirements in the three problem instances.

	Problem Instances		
	Alkmaar	*Rotterdam*	*Eindhoven*
Required drivers			
Complete *LSP*	4	10	8
Model 6.1	3	6	5
Model 6.4	3	6	5
Total walking time			
Complete *LSP*	11.0	27.1	26.6
Model 6.1	9.1	24.5	24.3
Model 6.4	9.0	24.2	24.1
Computation times			
Complete *LSP*	¡ 0.01	¡ 0.01	¡ 0.01
Model 6.1	5	11	25
Model 6.4	7	23	49

Table 6.2: A comparison of the driver assignments produced by the solution methods for instances with continuous driver shifts. The total walking time is listed in hours, and the computation time is in seconds.

For the total walking time of the train drivers we do see a small improvement when drivers can join train movements as passengers. However, the number of train movements with more than one driver in the solutions of Model 6.4 ranges from one in the *Alkmaar* instance to three in *Rotterdam*. The total walking time is reduced by at most twenty minutes when compared to the solutions of Model 6.1, suggesting that few beneficial opportunities for passengers are available in the shunting plans produced by the local search on these instances. Both MIP models do result in solutions with significantly less time spent on foot by the drivers than in the driver schedules produced by the list scheduling policy.

The stronger performances of the exact solution methods come at the cost of computation times that are several orders of magnitude larger than those of the list scheduling heuristic. With computation times of less than one minute, both MIP models can be used to find train driver assignments for individual shunting plans in practice. However, the exact methods are too slow to replace the list scheduling policy as the staff assignment sub-routine of the local search approach presented in Chapter 2. A local search algorithm requires an efficient evaluation of the candidate solutions to achieve good performance. Out of these three only the list scheduling heuristic, which finds feasible solutions in a few milliseconds, meets this criterion.

6.5.2 Non-identical driver shifts

When we focus on the train driver assignment problem with eight hour shifts, the instances require more time to solve with the proposed exact methods. Table 6.3

	Problem Instances		
	Alkmaar	*Rotterdam*	*Eindhoven*
Required driver shifts			
Complete *LSP*	10	26	18
Model 6.5	8	16	—
Model 6.5’	8	16	15
Computation times			
Complete *LSP*	¡ 0.01	¡ 0.01	¡ 0.01
Model 6.5	158	34	1800*
Model 6.5’	5	8	45

Table 6.3: A comparison of the driver assignments produced by the solution methods for instances with non-identical shifts. The total walking time is listed in hours, and the computation time is in seconds. Model 6.5 did not solve the *Eindhoven* instance within 30 minutes.

lists the results of the list scheduling heuristic and Models 6.5 and 6.5’ on the three instances. We set the maximum computation time of the exact methods to thirty minutes. The feasible solution spaces of the two MIP formulations are identical. However, as Model 6.5’ extends Model 6.5 with additional constraints, for the LP-relaxation we have that the solution space of Model 6.5’ is smaller than that of Model 6.5. Model 6.5 failed to solve the *Eindhoven* instance in the allotted time.

As with the instances with full-day shifts, these results illustrate that the exact methods can improve the train driver assignment significantly. In the optimal assignments the number of train driver shifts necessary to operate all train movements is reduced by ten to almost forty percent when compared to the heuristic solutions.

Due to the many shifts in these more realistic instances, the computation time of Model 6.5 increases rapidly. The model could not be solved within thirty minutes for the largest instance. In contrast, all instances are solved in less than one minute with Model 6.5’. The results show that the additional constraints in Model 6.5’ help the solver to find better bounds on the objective function. For example, with the *Eindhoven* instance the objective of the LP relaxation of Model 6.5 is 8, and after 30 minutes the (rounded) best objective bound is 10. In comparison, the objective of the LP relaxation of Model 6.5’ is 13, which results in a significantly smaller gap between the objective bound and the optimal solution cost.

In addition to the direct MIP formulations we have also tested the branch-and-price method on these three instances. However, the branch-and-price method did not solve any of the instances within the allotted 30 minutes.

6.6 Conclusion

In this chapter we proposed multiple exact solution methods based on mixed integer programming for the train driver assignment problem that arises in the train unit shunting problem at railway yards. For the problem variant with continuous availability of the drivers we formulated MIP models that maximize slack-based robustness measures as well as a model that allows train drivers to board other scheduled train movements to move more efficiently on the railway yard. For the case that the availability of the train drivers is restricted to shifts with a duration smaller than the full planning horizon we proposed a direct MIP formulation and a branch-and-price model.

To study the quality of the train driver assignments produced by the list scheduling policy introduced in Chapter 5, we compared these solutions with the assignments constructed with the exact MIP models. We solved three real-world instances of week days at major train stations in the Netherlands with both the MIP models and the list scheduling heuristic. The results showed that the optimal solutions required significantly fewer train drivers to feasibly operate all trains in the shunting plans, allowing the surplus of drivers to handle disruptions in the time table that might occur during operations or even a reduction in the number of scheduled drivers.

Furthermore, we have investigated the impact of allowing multiple train drivers on the same train movement to reduce their walking time. The results on the three problem instances suggested that the benefit of this option is limited, with at most a two percent reduction in total walking duration of the train drivers. The passenger extension might be preferable for instances with distant railway yards, since the overhead in computation time with respect to the basic model is small. Note that assigning train drivers as passengers of train movements results in additional temporal constraints between the movements, which might have a negative effect on the robustness of the driver assignment.

For the instances with train driver shifts of eight hours the direct MIP model constructed solutions in which up to forty percent fewer drivers were required to operate all train movements. By strengthening the LP relaxation of the MIP model with additional constraints the model was solved in less than sixty seconds. Our branch-and-price algorithm could not solve these instances within thirty minutes.

The computation times of solving the direct MIP models of the different problem instances were at most one minute in all cases. In contrast, the list scheduling policy constructs solutions in milliseconds. Although the MIP models cannot be solved efficiently enough to completely replace the list scheduling policy as the staff assignment sub-routine in the local search proposed in Chapter 5, the exact approaches do construct much better solutions. To improve the quality of the train driver assignments the MIP models can be used as a post-processing step after the local search procedure. Furthermore, applying the MIP models in parts of the local search might be beneficial and is a topic for further research.

Although minimizing the number of train drivers assigned in the shunting plans and maximizing the sum of slack times are some approaches to creating robust solutions, further improvements to the robustness could be made by integrating other robustness measures discussed in Chapter 3 in the objective of the train assignment problem. A direction for future research is to extend the MIP models proposed in this chapter with linearized versions of the robustness measures that are strongly correlated with the robustness of shunting plans.

Another area for further research is the inclusion of other staff scheduling constraints that are important in practice in the exact models. For example, the train drivers have a mandatory break during their shift. The regulations specify that this break must start at least two hours after the start of the shift, and be completed before the last two hours of the shift.

Chapter 7

Conclusions

The continuing growth in rail utilization across the globe leads to increasingly complex logistic challenges in the planning and operation of major hubs in the railway network.

In this thesis, we studied the shunting process at railway yards in these rail hubs, and developed a novel decision support framework to construct feasible and robust shunting plans. We first presented a local search approach that is capable of constructing feasible solutions for the train unit shunting problem with service scheduling in real-life instances of railway yards, where we assumed that there is always sufficient staff available. The next topic that we investigated in this thesis was the resilience of schedules to frequently occurring disruptions. We quantified the predictive power of surrogate robustness measures in shunting plans, and extended the objective function of the local search algorithm with some of the well-performing robustness measures to guide the local search to robust solutions. The last part of this thesis was centered around the staff assignment sub-problem of the shunting problem. We proposed and evaluated both heuristic and exact methods for this sub-problem.

7.1 Summary of Scientific Contributions

We started in Chapter 2 with a definition of the *Train Unit Shunting Problem with Service Scheduling (TUSPwSS).* This problem consists of six sub-problems: assigning train units arriving at the railway yard to departure time-slots; splitting and combining trains to achieve the required departure compositions; scheduling the required service tasks of the trains; parking trains on the available tracks; routing the trains safely over the yard; assigning the correct staff members to all tasks. Different researchers have studied MIP models to solve the shunting problem. We proposed a local search that uses a partial-order-based solution representation and exploits this structure to efficiently generate and assess neighbor solutions and hence efficiently search the solution space. Our local search contains all the first five sub-problems, which turns out to be very difficult with a MIP. We concluded that our local search algorithm is the first method capable of solving real-world instances provided by the rail operator NS. Moreover, the local search algorithm outperforms an existing MIP-based heuristic on realistic instances.

We shifted our focus from finding feasible shunting plans to constructing solutions that are resilient to commonly occurring operational disturbances such as delays in Chapters 3 and 4. In particular, we studied the applicability of surrogate robustness measures, which are characteristics of the behavior of a schedule in simplified stochastic models. We started in Chapter 3 by studying the properties of surrogate robustness measures in relation to different characterizations of robustness in stochastic machine scheduling problems.

With the insights obtained in this study we returned in Chapter 4 to the Train Unit Shunting Problem, where we quantified the predictive power of the surrogate robustness measures in shunting plans. We concluded that the strongest predictors of robustness were the robustness measures based on either the normal approximation method or the minimum slack. Based on these results we extended the objective of the local search with these robustness measures to guide the local search to robust solutions. We showed that adding the normal approximation robustness measure to the objective of the local search results in shunting plans that are significantly more robust at the cost of a small overhead in computation time.

In Chapter 5 we proposed two heuristic methods to solve the staff scheduling problem at shunting yards. The first heuristic assigned staff members and start times to the activities using a list scheduling policy. The second heuristic was a two-phase approach in which we first created a staff assignment, and then assigned start times to all activities. We concluded that the local search presented in Chapter 2 with the list scheduling policy as a sub-routine can solve real-world problem instances of the complete TUSPwSS including staff scheduling. To the best of our knowledge this algorithm is the first method to solve TUSPwSS instances of such size.

In Chapter 6 we proposed several exact formulations of the staff assignment sub-problem that are based on mixed integer linear programming and column generation. To evaluate the quality of the heuristic solutions, we compared the staff schedules constructed by the heuristics with the optimal schedules. We showed that the exact methods produce staff assignments that require significantly fewer staff members, in particular in the instances where staff members work in eight hours shifts instead of continuously throughout the planning horizon. Since the exact methods required up to one minute to solve the instances, we concluded that these mathematical models are computationally too demanding to replace the list scheduling policy as the staff assignment sub-route of the local search algorithm. However, the exact solutions methods can be applied as post-processing to improve the shunting plans constructed by the local search approach.

7.2 Practical Innovations

One of the goals of this thesis was to develop planning methods that can be applied in practice by passenger railway operators to support their shunting process. In co-operation with NS we made significant and impactful steps towards achieving this goal. Together with researchers and software developers at NS we transformed the scientific contributions summarized in the previous section into tangible software solutions for the logistic challenges that arise in passenger railway shunting.

The prime example of our collaboration with NS is the development of the decision support system *Hybrid Integrated Planner (HIP)*, which is based on the methods proposed in Chapters 2 and 5. The tool searches for feasible solutions to the complete Train Unit Shunting Problem with Service Scheduling at the railway hubs operated by NS, and returns the best solution found. The resulting plan can be viewed and edited by the human planners in the graphical user interface of the tool. In Chapter 2 we discussed a pilot study that has been conducted by NS with the decision support system on two railway hubs in the Netherlands. The goals of the pilot study were to evaluate the performance of HIP as well as the experiences of the planners using the decision support tool. The pilot was a success: the solution method solved most of the real-world instances and the feedback of the planners was positive. Although the tool failed to find feasible solutions in some cases, the human planners were able to construct feasible shunting plans much faster with the support of HIP than when they had to start from scratch. In other cases the tool seemed to find plans that could accommodate more train units on the yard. Moreover, the decision support system can search for shunting plans outside working hours of the planners, thereby further shortening the lead time of constructing plans. The planners involved in the pilot study responded positively to the decision support tool in general, and particularly appreciated the flexibility of the tool with regard to adding new constraints and planner preferences. Due to the successful outcome of the pilot study, NS has decided to implement HIP in the planning process of all railway hubs in the Netherlands in the coming years. With the shorter lead times due to HIP, the planning process can start closer to the day of implementation. This improves the accuracy of the data available to the planners, and thus reduces the need for plan adjustments before or during operations.

NS is in the top three of the world with respect to punctuality, but on a highly utilized rail network the occurrence of delays is inevitable. In Chapters 3 and 4 we showed that the resilience of the shunting plans to such disruptions can be improved significantly by embedding robustness measures in the objective function of our local search method. Moreover, the small computational overhead imposed by these robustness measure allows real-world problem instances at railway yards to be solved in acceptable time. By constructing and implementing robust shunting plans, railway operators such as NS can reduce the probability that disturbances in the shunting process result in delays in the timetable, which in turn helps to further improve their punctuality and passenger satisfaction.

Regarding the topic of the staff assignment, there is interest within NS to improve the rostering of the shifts of the train drivers at the railway hubs, i.e., the number of shifts and when the shifts start. The train driver shifts are determined weeks before the shunting plan, because the roster of the train drivers must be published on time. The goal is to schedule the shifts such that the number of available train drivers is sufficient to handle most disruptions, without having too many idle train drivers. These shift schedules are currently evaluated by the planners. We are working with NS to improve this evaluation process by using HIP, which contains the list scheduling policy proposed in Chapter 5, to estimate the quality of the shift schedules. A line of research that we are currently pursuing is to first generate problem instances and then determine the likelihood of finding feasible shunting plans and staff assignments with HIP for these instances with the giving train driver schedule. The exact methods proposed in Chapter 6 can be used to quantify the surplus of train drivers in the solutions.

7.3 Further Research

It is infeasible to study all intricacies and practical extensions of a challenging problem such as TUSP with service and staff scheduling in one thesis. We conclude by discussing some open problems and directions for further research related to the topics of this thesis.

With heuristic methods such as the local search approach proposed in Chapter 2 there can be problem instances that the solution methods fails to solve, but which might be solvable by other methods. Determining whether these instances are infeasible or not is still an open problem. In the absence of an exact method that can solve non-trivial instances, the next best approach is to solve relaxations of the TUSPwSS. The individual matching, service scheduling and parking sub-problems can be solved efficiently, but often provide weak bounds of the complete TUSPwSS problem. The planning of the train movements appears to be a bottleneck in many instances, but defining and solving a relaxation of TUSPwSS that preserves the train routing sub-problem remains an open problem.

Extending the scope of the TUSP from a single shunting yard to a major train station with multiple nearby railway yard poses several new challenges. Although we have adapted our local search algorithm in Section 2.5 to cope with the new elements introduced in the extended scope, experiments show that the local search struggles to find feasible solutions when the assignment of trains to the different railway yards is not properly balanced in the initial solution. This is mainly caused by the lack of local search operators that move all activities of a single train from one railway yard to another, and the large number of railway network movements that obstruct traffic between the yards. Solving TUSPwSS instances that contain multiple railway yards is an interesting topic for future research.

In Chapters 3 and 4 we restrict the sources of uncertainty to the arrival times of trains and the duration of service tasks and train movements. However, there are many more causes of disturbances in practice. Trains can arrive in a different order, or their compositions are not as planned. Additional trains might arrive if there is a disruption on the railway network, or part of the yard infrastructure can go out of service unexpectedly.

Achieving a resilient shunting process that adapts to these events requires further research in the directions of robustness measures for the smaller disturbances and recovery strategies for the large disruptions that cannot be mitigated in the shunting plan.

Our assumption in Chapters 5 and 6 that all personnel is available during their entire shift does not hold in practice. For example, staff members are entitled to a one-hour break, which has to be scheduled near the middle of the shift. Including these constraints in the heuristics proposed in Chapter 5 as well as in the exact methods of Chapter 6 has to be explored in future research to ensure that the staff assignments produced by HIP are in accordance with the regulations.

Bibliography

van den Akker, J. M., H. Baarsma, J. L. Hurink, M. S. Modelski, J. J. Paulus, D. A. Reijnen I. C .and Roozemond, and J. Schreuder (2008). Shunting passenger trains: getting ready for departure. *Proceedings of European Study Group Mathematics with Industry* 63.

van den Akker, J. M., J. A. Hoogeveen, and J. W. van Kempen (2012). Using column generation to solve parallel machine scheduling problems with minmax objective functions. *Journal of Scheduling* 15.6, pp. 801–810.

van den Broek, R. W., J. A. Hoogeveen, and J. M. van den Akker (2018). How to measure the robustness of shunting plans. *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

— (2019). Finding robust shunting plans. *10th Triennial Symposium on Transportation Analysis, (TRISTAN 2019)*.

— (2020). Personnel scheduling on railway yards. *20th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

van den Broek, R. W., J. A. Hoogeveen, J. M. van den Akker, and B. Huisman (2021). A local search algorithm for train unit shunting with service scheduling. *Transportation Science* 56.1, pp. 141–161. DOI: 10.1287/trsc.2021.1090.

Bürgy, R., H. Gröflin, and D. N. Pham (2011). The flexible blocking job shop with transfer and set-up times. *Journal of Combinatorial Optimization* 22.2, pp. 121–144.

Canon, L.-C. and E. Jeannot (2010). Evaluation and optimization of the robustness of DAG schedules in heterogeneous environments. *IEEE Transactions on Parallel and Distributed Systems* 21.4, pp. 532–546.

Černỳ, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications* 45.1, pp. 41–51.

Chtourou, H. and M. Haouari (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering* 55.1, pp. 183–194.

Dechter, R., I. Meiri, and J. Pearl (1991). Temporal constraint networks. *Artificial Intelligence* 49.1-3, pp. 61–95.

Dell'Amico, M. and M. Trubian (1993). Applying tabu search to the job-shop scheduling problem. *Annals of Operations Research* 41.3, pp. 231–252.

Al-Fawzan, M. A. and M. Haouari (2005). A bi-objective model for robust resource-constrained project scheduling. *International Journal of Production Economics* 96.2, pp. 175–187.

Flake, G. W. and E. B. Baum (2002). Rush Hour is PSPACE-complete, or "why you should generously tip parking lot attendants". *Theoretical Computer Science* 270.1, pp. 895–911.

Freling, R., R. M. Lentink, L. G. Kroon, and D. Huisman (2005). Shunting of passenger train units in a railway station. *Transportation Science* 39.2, pp. 261–272.

Gallo, G. and F. Di Miele (2001). Dispatching buses in parking depots. *Transportation Science* 35.3, pp. 322–330.

Haahr, J. T., R. M. Lusby, and J. C. Wagenaar (2015). A comparison of optimization methods for solving the depot matching and parking problem. *ERIM Report Series Research in Management* ERS-2015-013-LIS.

Hart, P. E., N. J. Nilsson, and B. Raphael (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4.2, pp. 100–107.

Hazır, Ö., M. Haouari, and E. Erel (2010). Robust scheduling and robustness measures for the discrete time/cost trade-off problem. *European Journal of Operational Research* 207.2, pp. 633–643.

Jacobsen, P. M. and D. Pisinger (2011). Train shunting at a workshop area. *Flexible Services and Manufacturing Journal* 23.2, pp. 156–180.

Jorge Leon V .and David Wu, S. and R. H. Storer (1994). Robustness measures and robust scheduling for job shops. *IIE Transactions* 26.5, pp. 32–43.

Khemakhem, M. A. and H. Chtourou (2013). Efficient robustness measures for the resource-constrained project scheduling problem. *International Journal of Industrial and Systems Engineering* 14.2, pp. 245–267.

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi (1983). Optimization by simulated annealing. *Science* 220.4598, pp. 671–680.

Kobylański, P. and D. Kuchta (2007). A note on the paper by MA Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics* 107.2, pp. 496–501.

Kroon, L. G., R. M. Lentink, and A. Schrijver (2006). Shunting of passenger train units: an integrated approach. *Transportation Science* 42.4, pp. 436–449.

Lentink, R. M. (2006). Algorithmic Decision Support for Shunt Planning. PhD thesis. Erasmus University Rotterdam: School of Economics.

Lentink, R. M., P.-J. Fioole, L. G. Kroon, and C. van 't Woudt (2006). Applying operations research techniques to planning of train shunting. *Planning in Intelligent Systems: Aspects, Motivations, and Methods*, pp. 415–436.

Liaw, C.-F. (1999). A tabu search algorithm for the open shop scheduling problem. *Computers & Operations Research* 26.2, pp. 109–126.

Ludwig, A., R. H. Möhring, and F. Stork (2001). A computational study on bounding the makespan distribution in stochastic project networks. *Annals of Operations Research* 102.1-4, pp. 49–64.

Mountakis, S., T. Klos, and C. Witteveen (2015). Temporal flexibility revisited: maximizing flexibility by computing bipartite matchings. *ICAPS*, pp. 174–178.

Nadarajah, S. and S. Kotz (2008). Exact distribution of the max/min of two Gaussian random variables. *IEEE Transactions on Very Large Scale Integration (VLSI) systems* 16.2, pp. 210–212.

Onomiwo, Z., J. A. Hoogeveen, M. A. van den Akker, and R. W. van den Broek (2020). Disruption management on shunting yards with tabu search and simulated annealing. MA thesis. Utrecht University.

Passage, G. J. P. N., J. A. Hoogeveen, and J. M. van den Akker (2016). Combining local search and heuristics for solving robust parallel machine scheduling. MA thesis. Utrecht University.

Wilson, M. (2016). Robust Scheduling in an Uncertain Environment. PhD thesis. TU Delft.

Wilson, M., T. Klos, C. Witteveen, and B. Huisman (2014). Flexibility and decoupling in simple temporal networks. *Artificial Intelligence* 214, pp. 26–44.

Nederlandse Samenvatting

Door economische en sociale ontwikkelingen is de vraag naar spoorvervoer hard gegroeid in de afgelopen decennia. De toenemende behoefte aan duurzaam transport maakt dat deze trend zich in de nabije toekomst zal voortzetten. Een grote uitdaging hierbij is dat de vraag naar treinvervoer sneller groeit dan de uitbreiding van het spoornetwerk, aangezien het bijleggen van sporen vraagt om grote investeringen en moeilijk te realiseren is in stedelijk gebied. Om mee te gaan met het groeiende aantal treinreizigers wordt er door overheden en spoorvervoerders dan ook volop ingezet op een efficiënter gebruik van de bestaande spoorinfrastructuur.

De toenemende drukte op het spoor leidt tot allerlei uitdagende planproblemen. Spoorwegmaatschappijen maken steeds vollere dienstregelingen en breiden hun treinvloten uit om alle ritten uit te voeren. Dit zorgt weer voor extra druk op de planning van treinonderhoud, -bewegingen en -opslag.

De voor de reiziger zichtbare kant van de planning, zoals de dienstregeling, kent een rijke onderzoeksgeschiedenis. In het automatiseren van de dienstregeling zijn er in de laatste decennia grote stappen gezet door wetenschappers wereldwijd. Hierdoor zijn spoorvervoerders in staat om efficiënte dienstregelingen te maken met behulp van automatische beslissingsondersteunende systemen.

In tegenstelling tot deze ontwikkelingen in de planning op spoornetwerkniveau wordt het lokale logistieke proces rondom de stations — de knooppunten in het spoornetwerk — nog grotendeels handmatig gedaan. Een *knooppunt* bestaat uit een station met één of meer nabijgelegen rangeerterreinen. Deze rangeerterreinen worden gebruikt als parkeerplaats voor treinen die niet nodig zijn binnen de dienstregeling, bijvoorbeeld tijdens de daluren. Daarnaast kunnen treinen ook gereinigd en geïnspecteerd worden op de rangeerterreinen. Alle treinbewegingen en onderhoudsactiviteiten binnen een knooppunt worden vooraf gepland en vormen tezamen het *knoopplan*.

Het doel van het onderzoek in dit proefschrift is het ondersteunen en verbeteren van de planning van de knooppunten in het spoornetwerk, waarbij we alle aspecten van het planproces meenemen. Een tweede doel is het verhogen van de betrouwbaarheid van deze knoopplannen in de operatie door in de planning al rekening te houden met mogelijke verstoringen.

In het eerste deel van dit proefschrift richten we ons op de rangeer- en onderhoudsplanning binnen een rangeerterrein. In dit planningsprobleem arriveren treinen op het rangeerterrein volgens een dienstregeling. Evenzo is bekend wan-

neer er treinen vanaf het rangeerterrein weer terug moeten naar het station. Het doel van de planning is om zonder conflicten alle treinen te onderhouden, parkeren en rangeren gedurende hun aanwezigheid op het rangeerterrein. Hierbij heeft de planner de vrijheid om zelf te kiezen welke van de aanwezige treinen van de juist types aan de geplande vertrekken in de dienstregeling toegewezen worden. In Hoofdstuk 2 presenteren wij een planningsalgoritme voor het rangeer- en onderhoudsprobleem. Het algoritme is gebaseerd op *local search* en zoekt naar een conflictvrije oplossing door iteratief kleine aanpassingen te maken aan een plan. We laten zien dat dit algoritme realistische planproblemen op rangeerterreinen gebruikt door spoorwegmaatschappij NS kan oplossen. Voor zover wij weten is dit het eerste algoritme dat hiertoe in staat is. Verder tonen we enkele uitbreidingen van het algoritme om plannen te maken voor de volledige knoop, inclusief stationsgebied, in plaats van enkel losse rangeerterreinen. Op basis van deze uitkomsten heeft NS een project gestart om het planningsalgoritme in gebruik te nemen op knopen in Nederland.

In Hoofdstukken 3 en 4 verschuiven we onze aandacht naar de robuustheid van knoopplannen tegen alledaagse, kleine verstoringen. We beginnen in Hoofdstuk 3 met een overzicht van methodes die gebruikt worden voor het schatten van de robuustheid van oplossingen van algemene stochastische planningsproblemen met beperkingen in tijd en middelen. Het belangrijkste kenmerk van deze problemen is de onzekerheid in de uitvoering van de planning, waarbij verstoringen kunnen leiden tot een afwijking van het opgestelde plan. Aangezien tijdens de planning nog niet bekend is welke verstoringen zullen optreden, is het van belang om robuuste plannen te maken. Dat wil zeggen, we willen plannen die in staat zijn om verstoringen op te vangen zonder al te veel in te boeten in kwaliteit. Het definiëren en berekenen van de robuustheid van plannen is echter complex, waardoor veel onderzoek zich gericht heeft op het ontwikkelen van afgeleide *robuustheidsmaten*. Dit zijn planeigenschappen die kunnen dienen als surrogaat voor de robuustheid met vaak als voordeel dat ze efficiënter te berekenen zijn. We introduceren een aantal nieuwe robuustheidsmaten en bestuderen de verschillende eigenschappen van zowel bestaande als de nieuwe robuustheidsmaten.

We passen de opgedane inzichten in de robuustheidsmaten toe op de rangeer- en onderhoudsplanning binnen de rangeerterreinen in Hoofdstuk 4. Voor de robuustheid van rangeer- en onderhoudsplannen zijn we met name geïnteresseerd in de kans op vertraging bij treinen die vertrekken van het rangeerterrein. Deze treinen gaan immers weer starten binnen de dienstregeling, dus vertrekvertragingen vanaf een rangeerterrein leiden tot verstoringen waar de reizigers last van hebben. In dit hoofdstuk starten we met het vergelijken van de kans op vertrekvertraging, die we geschat hebben met een stochastische *discrete-event* simulatie, en de uitkomsten van de verschillende robuustheidsmaten. Het doel van deze vergelijking is om robuustheidsmaten te identificeren die sterk correleren met de kans op vertrekvertraging en zodoende geschikt zijn om snel te evalueren of een rangeer- en onderhoudsplan robuust is. Onze experimenten laten zien dat de beste voorspellers van de kans op vertrekvertragingen bij de onderzochte plannen bestaan uit robuustheidsmaten die gebaseerd zijn op benaderingen met normale verdelingen of de minimale speling in het plan.

De vervolgvraag in dit hoofdstuk is of we met behulp van de robuustheidsmaten ook robuuste plannen kunnen maken. Met het planningsalgoritme uit Hoofdstuk 2, die in de basis geen rekening houdt met robuustheid, hebben we plannen gegenereerd voor verschillende realistische instanties van planningsproblemen voor een rangeerterrein dat gebruikt wordt door NS. Voor een selectie van goedpresterende robuustheidsmaten hebben we de doelstellingsfunctie van het planningsalgoritme uitgebreid met sturing op de robuustheidsmaat, om zo voorkeur te geven aan robuustere plannen. Met deze robuustheidsmaatgestuurde varianten van het planalgoritme hebben we voor plannen gegenereerd voor dezelfde realistische instanties. Voor alle plannen hebben we de kans op vertrekvertraging geschat met behulp van een stochastische simulatiestudie. We laten zien dat het sturen op een robuustheidsmaat binnen het planalgoritme leidt tot significant robuustere plannen. In het bijzonder geeft het schatten van de robuustheid met behulp van normale verdelingen zeer robuuste rangeer- en onderhoudsplannen ten koste van een geringe toename in rekentijd van het algoritme.

In het laatste deel van dit proefschrift onderzoeken we het verrijken van het rangeer- en onderhoudsplan voor de rangeerterreinen met een personeelsplanning. Voor de verschillende treinactiviteiten op het rangeerterrein is personeel, zoals machinisten en monteurs, nodig en ook wie wanneer welke taak uitvoert moet vooraf gepland worden. Rangeerterreinen bestaan uit kilometers aan spoor, waardoor het efficiënt bundelen van taken noodzakelijk is binnen de personeelsplanning om grote looptijden te voorkomen. We presenteren in Hoofdstuk 5 twee heuristieken voor de personeelstoewijzing die geïntegreerd kunnen worden met het algoritme uit Hoofdstuk 2. Hiermee is het voor het eerst mogelijk om automatisch een volledige rangeer-, onderhouds- en personeelsplanning te maken voor rangeerterreinen en spoorknooppunten. We laten zien voor een verzameling van planproblemen met personeel op een rangeerterrein in Nederland dat het planalgoritme uit Hoofdstuk 2 samen met de bestpresterende personeelstoewijzingsheuristiek in staat is om 97% van de planproblemen op te lossen in gemiddeld drie minuten rekentijd.

In Hoofdstuk 6 bekijken we het toewijzen van machinisten aan rangeerbewegingen op knopen in meer detail. Dit deelprobleem van de algemene personeelstoewijzing is op veel knopen het meest uitdagende onderdeel. We introduceren verschillende exacte modellen gebaseerd op geheeltallig lineair programmeren, waarin we onder andere het benodigde aantal machinisten minimaliseren en onderzoeken wat de invloed is van het laten *passagieren* — meereizen met al geplande treinbewegingen — van machinisten op de totale looptijd en oplossingskwaliteit. Onze experimenten laten zien dat de kwaliteit van de optimale machinistentoewijzingen significant beter is dan die van de heuristieken uit Hoofdstuk 5. Daar staat tegenover dat de rekentijden van de mathematische modellen velen malen hoger liggen, waardoor deze exacte oplossingsmethodes ongeschikt zijn om te gebruiken binnen het planningsalgoritme uit Hoofdstuk 2.

Curriculum Vitae

Roel Wemarus van den Broek was born on May 2nd in Tiel, The Netherlands. In 2009, he received his preparatory scientific education (VWO) degree from the Konining Wilhelmina College in Culemborg.

In 2014, he obtained bachelor's degrees (cum laude) in both Mathematics and Computer Science from Utrecht University. In 2016, he completed the master programme Computing Science from the same university cum laude. His master's thesis was entitled 'Train Shunting and Service Scheduling: an integrated local search approach'. The thesis was supervised by Dr. Han Hoogeveen and Dr.ir. Marjan. In 2016, he started as a PhD student in the Department of Information and Computing Science at Utrecht University under the supervision of Dr. Han Hoogeveen, Dr.ir. Marjan van den Akker and Prof.dr. Hans Bodlaender. As of November 2020, he is working as a researcher and software developer at the R&D Hub Logistics department of NS.