Erasmus University Rotterdam

Erasmus School of Economics

Heuristics for Rolling Stock Rebalancing

Rolling Stock Rebalancing Problem

Master Thesis
MSc Operations Research and Quantitative Logistics

Author:
L. Leijten
(451632)

Supervisor:
D. Huisman

Company Supervisor:
J. Van 't Wout

Second Assessor:
R. Hoogervorst

Friday 29th January, 2021

Abstract

This thesis copes with the rolling stock rebalancing problem. This problem is correlated with the rolling stock circulation problem, which assigns rolling stock to trips of a timetable, such that the capacity of the assigned train units matches the expected number of passengers. In practice, the assignment of rolling stock is calculated for each day separately. As a result, the end-of-day inventory of rolling stock is in general not equal to the start-of-day inventory of the succeeding day. The goal of the rolling stock rebalancing problem is to dissolve these arisen unbalances.

First, a method is proposed which re-uses a mathematical formulation of another paper. Secondly, a heuristic approach will be explained which consists of two self-contained methods. The first method uses a time-dependent graph to illustrate all possibilities for a train unit to traverse from one station to another. The corresponding shortest path of the graph illustrates the least-cost action to dissolve an unbalance between two stations. Then, simulated annealing is introduced, where the neighbourhoods randomly alter the initial schedule such that the number of (weighted) unbalances decreases. Both these methods are used in an overall, heuristic solution approach which iteratively searches for the best solution value.

The proposed solution approaches are tested on realistic instances of the Dutch Railways (NS). The results illustrate that the proposed solution approaches provide decent results. The computation time remains low, even for larger instances.

Contents

1 Introduction

Nederlandse Spoorwegen (Dutch Railways; NS) is the principal operator of the Dutch rail network. In 2019, over 1.3 million passengers used the train as daily transport and these passengers travelled a total of 17.5 billion kilometres over a year. NS shares the train tracks with other railway operators, such as Arriva, Connexxion and Keolis. Nevertheless, NS operates over 80% of the rail network of the Netherlands. Managing and controlling such a complex railway network and a large fleet of train units can be very challenging, especially when you consider that the Dutch railway network is the third busiest after Switzerland and Japan (UIC, 2015).

Creating a timetable, allocating rolling stock and assigning crew to such a dense network is a complicated problem. The complexity of this puzzle is extremely difficult and cannot be solved at once. Therefore, this puzzle is split into smaller problems which are easier to solve separately. The most commonly used subproblems are timetabling, train platforming, assigning rolling stock and crew scheduling. The first step is timetabling, which is the decision upon the routes and their respective arrival and departure times at every station. Secondly, during the train platforming, each trip gets its entrance and exit route specified such that each platform is either occupied or available at any point in time. Subsequently, the rolling stock is assigned to every trip such that the capacity of the rolling stock exceeds the expected number of passengers. Finally, the crew is assigned to a duty, such that every passenger train is authorised by a predetermined number of conductors and train drivers.

In this thesis, we focus on the rolling stock circulation problem, or, more specifically, the rolling stock rebalancing problem. The main goal of the rolling stock circulation problem is to assign available rolling stock to routes of a given timetable such that the capacity of the selected train units matches the expected number of passengers. Rolling stock units can be split from or combined with each other, such that there is a better match between the supply of train units and the demand of expected passengers. Meanwhile, the costs should be minimised and the service should be maximised. The main 'representatives' of these objectives are the carriage kilometres and the expected number of seat shortages. These are just the main restrictions and objectives of the rolling stock circulation problem. Later on, more constraints will be discussed and a better interpretation of the objective function will be given.

Currently, NS has access to a tool that solves the rolling stock circulation problem. The tool is called *Tool voor de Aanpassing van de Materieelinzet* or, abbreviated, TAM. The mathematical formulation of Fioole et al. (2006) is used as the basis for this tool. In general, TAM is particularly used to construct a rolling stock roster for a generic week and special days, such as (bank) holidays. It requires a lot of data to function, such as the details of all rail lines, the characteristics of the rolling stock and other conditions like shunting possibilities and forbidden compositions. Then, TAM provides us with a range of options to suffice our preferences, for instance, by choosing a subset of available rolling stock or by changing the weights of the objective function. With this information, TAM returns a feasible rolling stock roster.

As already mentioned, the mathematical model of Fioole et al. (2006) is used as the foundation of TAM. The paper deals with multiple interesting aspects of the rolling stock circulation problem, such as combining and splitting of rolling stock, the order of train units within their composition and it optimises over a multi-objective function. Unfortunately, the method described in this paper is focused on solving the rolling stock circulation problem for only one day. This leads to complications when we want to establish the rolling stock schedule for two (or even more) consecutive days. As a result, the computation

times explode because dealing with a longer time horizon leads to a larger and, therefore, a more complex problem. Solving the rolling stock schedule for each day separately, as is currently done, leads to multiple unbalances in rolling stock between the end of a day and the beginning of the succeeding day. This thesis will focus on solving the so-called rolling stock rebalancing problem which dissolves these arisen unbalances.

Until now, other approaches do generally not pay attention to unbalances in rolling stock. When papers refer to unbalances, the proposed methods are genuinely ineffective and/or the instances are relatively small in size. In the latter case, computation times seem low, but when the same solution approach is applied to larger instances, the running times explode. Consequently, considering the complex and large structure of the Dutch rail network, effective methods are necessary to come up with a decent solution within reasonable computation time. This thesis will primarily focus on methods to tackle the rolling stock rebalancing problem. The goal is to dissolve all unbalances by changing the initial schedules. Alternatively, more expensive empty movements can be used to dissolve unbalances. The total costs of the balanced schedule must be as low as possible. Here, both the quality of the solution and, especially, the computation time are crucial factors of a decent solution approach.

In this thesis, multiple solution approaches will be described to tackle the rolling stock rebalancing problem. First, an exact solution approach is suggested, which re-uses the mathematical formulation of Fioole et al. (2006). A time window must be selected which covers at least a fraction of both days. Then, this method re-optimises all trips within a selected time window.

Next, a heuristic solution approach is described, which consists of two self-contained solution approaches. The first method uses a time-dependent graph to represent all possibilities a train unit can traverse from one station to another. Then, a path generation approach iteratively generates paths from the time-dependent graphs. Each iteration, a linear program is solved and, if feasible, all unbalances are dissolved. Secondly, simulated annealing uses neighbourhoods which are randomly selected. The neighbourhoods alter the initial schedules, where the goal is to decrease the number of unbalances. Ultimately, these two methods are used in an overall, heuristic solution approach. An iterative process is used to first apply simulated annealing to the initial schedules. Afterwards, the remaining unbalances are dissolved by using path generation. The process is repeated such that this solution approach optimally exploits the diversification of simulated annealing.

The remainder of this thesis has the following structure. Chapter 2 describes the problem itself in more detail. First, a general description of the rolling stock circulation problem is introduced, including any relevant constraints and the objective function. Then, this chapter describes the relevant basics of the rebalancing problem. Chapter 3 provides a literature review dealing with rolling stock rebalancing. Because the literature regarding this specific problem is relatively limited, relevant articles about the rolling stock circulation problem are also reviewed. Then, the notation of the rolling stock circulation and rebalancing problem is clarified in Chapter 4. Afterwards, the proposed methods are described in respectively Chapter 5 and Chapter 6. The first method re-uses the mathematical formulation of another paper to come up with a schedule without any unbalances. The heuristic approach of Chapter 6 outlines two distinctive methods, which can be used separately and/or merged into one overall heuristic approach. Then, Chapter 7 reveals and analyses the results of all methods. The following Chapter 8 draws the conclusions. Lastly, Chapter 9 outlines limitations of the proposed solution approaches and provides recommendations for further research.

2 Problem Description

In this chapter, first, the rolling stock circulation problem is explained in Section 2.1. Here, multiple constraints and the idea of the multi-objective function are clarified. Afterwards, Section 2.2 introduces the rolling stock rebalancing problem. In this section, it will be explained how unbalances arise and what actions can be taken to dissolve the unbalances. Lastly, Section 2.3 provides a brief overview of the current procedure of NS for calculating the rolling stock schedule.

2.1 Rolling Stock Circulation Problem

A brief explanation of the rolling stock circulation problem is the assignment of train units to so-called trips such that the capacity matches the expected number of passengers, while at the same time minimising the specified objectives. The model formulation of Fioole et al. (2006) is still used by NS as the fundamental basis for determining the railway rolling stock schedule. Therefore, in this thesis, we try to stick to the formulation and notation of this mathematical model.

The rolling stock circulation problem is the next step after the timetable has been established. To efficiently use the timetable for the rolling stock circulation problem, the timetable is partitioned into trips. A trip is defined as a scheduled route between two stations, where the composition of train units cannot change between these two stations. Each trip requires a certain number of seats to exceed the estimated demand of passengers. One could take this restriction as a hard constraint, but seat shortages can also be implemented in the objective function, which will be clarified later on. Especially during rush hours, the demand for seats is generally higher than outside the peak hours and, accordingly, the demand during rush hours is sometimes difficult to meet with the available rolling stock.

The composition of one trip could consist of multiple and different types of train units. Each train unit has a prespecified capacity and by choosing different train units, the supply and demand for seats can be matched. In general, the capacity of a composition should exceed the estimated demand of passengers, especially outside rush hours. Between trips, train units can be coupled to and/or decoupled from other carriages. By changing the composition between trips, the seat surplus for less occupied train trips is minimised.

By matching the supply and demand as much as possible, energy costs are minimised and spare train units can be used to perform maintenance. However, the downside of performing many (de)coupling movements is that these operations are time-consuming. Furthermore, if time allows executing these movements, the outcome is a less robust roster. For example, assume one trip with two train units arrives at station A and subsequently two trips leave from station A, where both trips require only one train unit. A simple solution would be to split the composition of two train units at station A such that both succeeding trips can use one of the separated carriages. However, if the trip towards station A has some delay, it is plausible that the decoupling activity cannot be performed in time. Hence, this could lead to delaying both departing trips or even to cancelling (one of) the trips. In conclusion, shunting movements may lead to a more efficient usage of the rolling stock while it simultaneously leads to a less robust schedule.

Nonetheless, the rolling stock circulation problem consists of many rules that need to be followed. The composition of train units cannot exceed the length of a train platform. If one does not acknowledge this

restriction, then this causes problems for passengers who cannot leave the train at the far left or far right exit. Although one could allow these situations, we assume that the length of a train can never exceed the length of a platform. A trip usually passes through multiple stations and, therefore, the maximum allowed length is the length of the shortest station of that trip. Besides the length of a train station, some locations have limited space for shunting possibilities. For a valid roster, shunting movements at these stations could be limited or even disallowed. Moreover, a similar procedure could also apply, when the train station is relatively dense, due to the high number of arrivals and departures.

Furthermore, one could disallow specific compositions (for a certain trip). NS has multiple types of train units, but not all train units can be connected to each other. Some train units can never be coupled to each other, while other combinations are only disallowed for certain trips. Lastly, a station has a limited capacity in terms of space for train units. Unused train units are temporary stored at the shunting yard of a train station. The storage of a station has a certain capacity and this capacity cannot be exceeded.

The objective consists of three major aspects: costs, service and robustness. First of all, the costs can be expressed as operational costs which occur by driving the rolling stock. If a carriage drives more kilometres, it has direct costs of its usage e.g. traction power. Besides, if a train unit performs more kilometres, preventive maintenance is required more early. These costs can be expressed as the number of carriage kilometres.

Secondly, the service towards the passengers should be sufficient. The main component of service for this problem is seat capacity. If the estimated number of passengers exceeds the train capacity, passengers are forced to stand during their journey. Especially during rush hours, this is a common problem. One could argue that the train capacity should always exceed the number of passengers, but it is questionable whether this is feasible and/or efficient. Another approach is to include the seat shortage into the objective function. This is done by multiplying the expected number of seat shortages by the length of the respective trip, that is seat-shortages kilometres. In this thesis, seat shortages are not implemented as a hard constraint, but as seat-shortages kilometres in the objective function.

Lastly, the robustness of a schedule is also an important factor. We already mentioned that having many shunting movements will lead to a less robust roster, which is sensitive to delays and disruptions. The reliability of the rolling stock schedule can be increased by having less coupling and decoupling activities. Moreover, these activities also lead to direct costs, because a crew is required to perform these movements. For this reason, the number of shunting movements is minimised in the objective function. Note however that shunting movements may increase efficiency and better service, because the capacity of train units can match the number of expected passengers in a better way. Therefore, including more or less shunting movements is a trade-off between efficiency and robustness.

2.2 Rolling Stock Rebalancing

In the previous section, the rolling stock circulation problem for a single day is explained. However, in this thesis, the focus lies on the rolling stock rebalancing problem. In general, if the rolling stock schedules of consecutive days are calculated separately, multiple unbalances between the rosters arise. Technically speaking, it is possible to compute a rolling stock schedule of multiple consecutive days without considering the days separately. However, the major disadvantage of such a method is that the computation time increases rapidly, or even exponentially. In practice, the rolling stock schedule is

calculated independently for every single day. The independent calculations are easier in terms of running time, but the main downside is that this leads to multiple unbalances. The so-called end-of-day inventory is in general not equal to the start-of-day inventory of the succeeding day. Here, the start-of-day and end-of-day inventory represent a list of all available train units and their respective location at respectively the start of a day or the end of a day. Each difference between the end-of-day and start-of-day inventory results in an unbalance.

An unbalance depicts a train unit whose destination station of a day does not equal the departure station on the succeeding day. Ultimately, the goal is to match every destination station with a departure station. There are multiple techniques to dissolve an unbalance. One could decouple the train unit earlier (or later) such that the initial destination station alters. Similarly, the departure station can also be changed by executing such a movement. As a last resort, a deadheading movement could be performed. A deadheading trip is an empty train which is transferred between two stations to dissolve an unbalance. Let us introduce our primary assumption: a deadheading trip is always realisable between two stations at night. One should compare the costs of (de)coupling movements and the additional costs of a longer composition to the expenses of a deadheading trip. Because a deadheading trip is always feasible, the costs of an empty movement can be recognised as an upper bound for dissolving an unbalance.

If there are multiple unbalances, the problem suddenly becomes more difficult. Instead of considering only one destination and one departure station, we look at multiple destination and departure stations. Then, instead of only considering the least-cost option to traverse one train unit from a destination station to a start departure, one also needs to determine which destination station will be connected to which departure station. Furthermore, the increasing number of different train units types makes the problem even more difficult. A schedule is considered as in balance, if all unbalances are either solved by adjustments to the initial schedule and/or by deadheading movements.

A station has a surplus of a type if the number of arriving units on a day is higher than the number of leaving units on the succeeding day. On the contrary, a station is in shortage of a type if the number of arriving train units is higher than the number of required units on the following day. Note that a shortage or surplus depends specifically on the rolling stock type. It is likely that a station has a surplus for one type as well as a shortage for another train unit type.

2.3 Current Process

In order to start solving the rolling stock circulation problem, NS provided the necessary input for this problem. The data set consists of all available train lines in the Netherlands used by NS and many regulations. Besides the estimated number of passengers, the input consists of shunting rules, the maximum length of trips, forbidden compositions (for a certain trip) and other remaining exceptions. Furthermore, NS scales trips into certain groups, which differ with how seat shortages are penalised. For certain trips, especially during rush hours, NS accepts that a certain number of x passengers stand during their journey. This shortage of seats is not penalised unless the number of passengers is higher than x. Lastly, the specifications of the available train unit types and the corresponding allowed compositions are required to solve this problem.

This data set is used as input for TAM and it gives us a range of options to satisfy our preferences. It is possible to use only a subset of all available train types and one could also manually change some of

the loaded configurations. Interestingly, TAM has not only an option to solve multiple days separately, but it can also compute the rolling stock schedule of multiple days integrated. However, as expected, the running times explode and this is not recommended, especially for larger instances.

When the loaded configurations and selected preferences meet to your satisfaction, TAM solves the rolling stock circulation problem for these options. When the tool has finished, it shows the created roster graphically. Here, one can see which trips have a shortage (or a surplus) of seats. Furthermore, deadheading trips are highlighted and one can easily identify all shunting movements.

3 Literature

Fioole et al. (2006) describe the fundamental basis of the rolling stock circulation model currently used by NS. The paper describes a mixed-integer linear program with multiple objectives. The model includes (de)coupling movements between train units, the specific order of a train unit within its composition and a limited storage capacity for (temporary) unused rolling stock. More importantly, the paper takes a longer time horizon into account by setting the start-of-day and end-of-day inventory of the same day equal to each other. For the remainder of this thesis, the mathematical formulation of Fioole et al. (2006) is also referred to as the composition model. Nielsen et al. (2012) extend this model by changing the scheduling formulation into a rescheduling model. This paper also deals with unbalances by including deviations from a target inventory level to the mathematical formulation. If the inventory of the obtained solution differs from a predetermined inventory, then this deviation is penalised.

On the other hand, Haahr et al. (2014) present a path-based formulation, where a path corresponds to a sequence of trips that a single unit can perform. It is solved using a branch-and-price framework. Although the main focus lies on the rescheduling of train units during disruptions, this method can also be used for non-disturbed timetables. Haahr et al. (2016) compare the above-explained path-based formulation with the composition model of Fioole et al. (2006) for both scheduling and rescheduling instances. Regarding the scheduling part, the composition model has a significantly lower computation time compared to the one of Haahr et al. (2014), but including maintenance restrictions would be easier for the path-based formulation. The rescheduling extension of Nielsen et al. (2012) based on the composition model performs significantly better than the one described in Haahr et al. (2014).

That the so-called composition model is a 'revolutionary' model for the rolling stock circulation problem, also implies from the fact that other papers are based on or use their introduced model as a comparison to this paper. In addition, Nielsen (2011) compares the composition model with a so-called task model. Here, a trip consists of one or multiple tasks where a task represents a train unit assigned to that trip. Besides the rolling stock circulation problem, the paper also addresses uncertainty during disruptions and the response of passengers to these disturbances. Furthermore, Maróti (2006) describes multiple aspects of railway rolling stock planning. It provides solution methods for tactical planning (based on Fioole et al. (2006)) and operational planning. For the operational rolling stock planning, the paper fixes the rebalancing issues using a heuristic approach. Moreover, the paper addresses an interchange model and a transition model to solve the maintenance routing problem.

The article of Borndörfer et al. (2016) has a completely different angle of tackling the rolling stock circulation problem. It uses a so-called hypergraph which consists of a set of nodes, a set of hypernodes and a set of hyperarcs. A node is included for every individual trip and composition. A hypernode represents a feasible assignment of a vehicle configuration for a trip. The paper uses an LP-relaxation of a multi-objective integer programming formulation based on this hypergraph. Then, column generation is applied and a rapid branching method is used to obtain a cyclic planning for a standard week.

Cacchiani et al. (2010) present an integer-linear program, which is eventually used for an LP-based heuristic approach. Here, both maintenance and deadheading constraints are included in the heuristic. Lin and Kwan (2014) describe a complicated process, as it also includes the shunting process at train stations. It is a two-phase approach where it starts with a similar solution concept as in Cacchiani et al. (2010) to solve the train unit scheduling. On top of that, their article comes up with a mixed-

integer linear programming formulation to solve the railway shunting problem. It, furthermore, considers the rescheduling of train units due to possible delays or maintenance. Besides this approach, Lin and Kwan (2016) propose a path-based formulation which is based on a branch-and-price algorithm. Multiple restrictions like (de)coupling, deadheading and forbidden locations are included in the model. The paper solves the rolling stock circulation problem for the Scottish railways using multiple branching rules and adaptive node selection.

Zhong et al. (2019) split the rolling stock circulation problem into two stages. During the first stage, multiple feasible schedules are generated. The second stage checks whether the feasible schedules are compatible with respect to the maintenance constraints. Also, the paper of Wang et al. (2019) focuses on the maintenance constraints. Their approach is similar since they also separate the problem into two parts. However, they transform a given schedule into a feasible one by taking the maintenance constraints and the circulation into account.

Lusby et al. (2017) propose a solution technique which tackles the problem of rescheduling rolling stock during a disruption. In this paper, a path-based formulation is explained and it is solved using a branch-and-price algorithm. Tréfond et al. (2017) create rosters which tend to be more robust in terms of delay or other disruptions. It is a multi-step approach in which an integer linear program is used, which also incorporates maintenance constraints. Simulation is used to check the quality of the robustness for disruptions. Hoogervorst et al. (2019) also describe the rolling stock rescheduling problem. In this paper, the authors come up with a variable neighbourhood search heuristic, where they apply variable neighbourhood descent. Moreover, the paper explains three different neighbourhoods. The results show close-to-optimal solutions, although the number of cancellations seems to be relatively high.

Besides attempting to solve the rolling stock circulation problem to optimality, one could try to combine multiple aspects of the complete puzzle into one problem. That is exactly what the paper of Schöbel (2017) describes. This article integrates multiple parts of the railway planning into one integrated solution method. It combines line planning, timetabling and vehicle scheduling into one problem. The paper proposes a method using an eigenmodel, which is a graph where the nodes correspond to specified optimisation problems. A path through multiple nodes represents a sequence of multiple optimisation models. Cadarso and Marín (2012) also propose an integration of multiple stages into one overall method. This paper only combines the railway timetabling with vehicle scheduling, but it also includes constraints like maintenance and robustness. It is tested on the Spanish railway operator RENFE and this method leads to a more efficient roster. Haahr and Lusby (2017) integrate the rolling stock scheduling problem with the train unit shunting problem. In the paper, the authors compare two branch-and-cut algorithms and one branch-and-price method. Furthermore, they explain a local search technique which swaps units with each other.

Budai et al. (2010) deal with the rolling stock rebalancing problem. This paper uses a slightly different definition compared to this thesis, whereas the authors deal with a short-term planning period. Due to changing circumstances, the initial end-of-day inventory cannot be realised and, for this reason, new unbalances arise. First, the paper addresses a solution approach for solving one unbalance of a single train unit. Next, the paper describes two solution approaches based on the approach of solving one unbalance. These heuristics are respectively based on a greedy algorithm and an integer program. The rolling stock rebalancing problem is also addressed in Farina et al. (2018). This paper pays more attention to deadheading movements as it is integrated into the objective function. The authors propose a mixed-

integer linear program, which is tested on instances of the Dutch and Danish rail network. Within one hour of running time, the proposed method solves (most of) the instances to optimality.

From the described literature, it can be concluded that only a few papers have a remark about a planning horizon of multiple days or consider the rebalancing of rolling stock. Borndörfer et al. (2016) and Farina et al. (2018) show results with instances of at least two consecutive days, but the number of considered trips of the instances is relatively low. Furthermore, Fioole et al. (2006) mention to set the start-of-day balance equal to the end-of-day balance to generate a cyclic solution, but its efficiency is questionable. Nonetheless, most papers addressing unbalances consider dynamic unbalances that appear due to delays and/or other disturbances. Because of these disruptions, unbalances appear and the initial rolling stock roster is infeasible. The solution approaches, which solve the so-called rolling stock rescheduling problem, are in general relatively fast, because the unexpected disruptions and the respective unbalances need to be solved as quickly as possible to prevent further disruptions. Multiple solution methods are presented in the literature such as variable neighbourhood search (Hoogervorst et al., 2019), linear programming (Haahr et al., 2014) and (greedy) heuristics (Budai et al., 2010).

4 Problem Formulation

In this chapter, the notation, which will be used throughout this thesis, will be introduced. First of all, the mathematical formulation of the rolling stock circulation problem is presented in Section 4.1. Secondly, Section 4.2 explains the objective function in more detail. The objective function is used for the rolling stock circulation problem as well as for the rolling stock rebalancing problem. In Section 4.3, we describe how a feasible solution can be mathematically expressed. This feasible solution will be used for the rolling stock rebalancing problem, whose formulation is explained in Section 4.4.

4.1 Rolling Stock Circulation Problem

The formulation and notation of this section are based on the mathematical model described in Fioole et al. (2006). First, let us introduce the set of stations S. An element $s \in S$ is either the arrival $s_a(t)$ or the departure station $s_d(t)$ of a trip $t \in T$. A trip departs at a certain time $\tau_d(t)$ from station $s_d(t)$ and arrives at a station $s_a(t)$ at time $\tau_a(t)$. Let ℓ_t denote the number of kilometres of trip $t \in T$. For each (passenger) trip, a composition of at least one train unit must be chosen. Let us introduce the set of train unit types $m \in M$. Train unit types differ in capacity, length ℓ_m and the number of carriages c_m.

A composition is any feasible combination of train units and it can consist of different train unit types m. Every trip has a particular set of allowed compositions $P_t, t \in T$. Within the set of allowed compositions, the order of train units types is critical. Assume there are two different types of train units: $m_1, m_2 \in M$. Then, for any arbitrary trip $t \in T$, the set of allowed compositions presumably consists of the following elements: $\{m_1, m_2, \{m_1, m_1\}, \{m_1, m_2\}, \{m_2, m_1\}, \{m_2, m_2\}\}$. Note that $\{m_1, m_2\}$ and $\{m_2, m_1\}$ are two unique elements, but there is only one element $\{m_2, m_2\}$. We do not distinguish between train units of the same type. However, we distinguish between $\{m_1, m_2\}$ and $\{m_2, m_1\}$, because the shunting possibilities could be limited. At some stations, it is only allowed to decouple a unit at the rear (or the front) of the train. The order of the composition is therefore crucial to determine which rolling stock is decoupled from the initial composition.

Besides taking the order of a composition into account, the set of allowed compositions P_t also bears the intended length in mind. If the length of a certain composition $p \in P_t$ exceeds the maximum length of a trip, then such a composition will be excluded from the set of allowed compositions. Let $n_{m,p}$ denote the number of train units of type $m \in M$ of composition $p \in P$. If the maximum length of a trip t is denoted as ℓ_t, then it must hold that $\sum_{m \in M} \ell_m \, n_{m,p} \leq \ell_t$ for any composition $p \in P_t$.

To implement the shunting possibilities, let $v(t)$ be the predefined successor trip of trip t. Then, both trip t and trip $v(t)$ have a set of allowed compositions, respectively P_t and $P_{v(t)}$. However, in general, one can not arbitrarily pick a composition $p_1 \in P_t$ and a composition $p_2 \in P_{v(t)}$. Because $v(t)$ is the successor trip of t, the number of shunting possibilities is relatively limited. The composition $P_{v(t)}$ is either equal to the composition P_t, lacks one or multiple decoupled train units compared to P_t or one or multiple units are coupled to composition P_t. Let Γ_t be the set of allowed compositions changes of trip $v(t)$ succeeding trip t. In other words, Γ_t defines a pair of compositions (p, p'), where $p \in P_t$ and $p' \in P_{v(t)}$. By defining the composition change in this way, limitations can be explicitly defined. One of the implementations of Γ_t is that it is prohibited to couple and decouple at the same station. Therefore, composition changes, which suggest a coupling and a decoupling movement, are eliminated from Γ_t.

If a decoupling activity is executed, a longer composition is split into two smaller ones. One of the smaller compositions must continue its journey by executing the successor trip $v(t)$. The other composition is temporarily stored at the arrival station of trip t. Before that stored composition can be used again, its re-allocation time $\varrho(s)$ should be respected. Depending on the station, a train unit requires a certain amount of time to be reallocated.

4.2 Objective Function

As already mentioned, the objective function is divided into several components. The main objectives are the carriage kilometres, seat shortages and the number of shunting movements. In this section, only these three objectives, which are also relevant to rolling stock rebalancing, will be explained in more detail. Yet, more objectives could be used such as seat excesses, the number of used train units or any modifications compared to a reference schedule.

The composition model is only used to generate a feasible schedule, which is eventually the input for the upcoming solution approaches. Knowledge about the mathematical formulation of the composition model is not necessary to understand the described methods and therefore, only relevant decision variables will be introduced. For more details about the mathematical formulation, we refer to the paper of Fioole et al. (2006). In this thesis, there are three essential components which define the objective function: carriage kilometres, seat shortages and shunting movements. Let the elements be abbreviated by respectively *CKM*, *SSK* and *SHM*. Then the objective function can be written as:

$$\min F(X, Z, N) = C_{CKM}\ CKM + C_{SSK}\ SSK + C_{SHM}\ SHM. \tag{1}$$

The relative importance of the three elements can be expressed by the weights, e.g. C_{CKM} serves as the weight of the number of carriage kilometres (CKM). Note that the objective function is based on three decision variables X, Z and N. Let us explain the components of the objective function and the appropriate decision variables in more detail.

Firstly, CKM represents the total number of carriage kilometres. Introduce decision variable $N_{t,m}$, which represents the number of train units of type $m \in M$ that are used for trip $t \in T$ of a given schedule. Hence, the decision variable must be an integer: $N_{t,m} \in \mathbb{Z}^+$. Recall that ℓ_t denotes the number of kilometres of trip t and c_m describes the number of carriages of composition m. Then, the total number of carriage kilometres can be written as:

$$CKM = \sum_{t \in T} \sum_{m \in M} \ell_t\ c_m\ N_{t,m}. \tag{2}$$

Second of all, SSK expresses the number of kilometres where the number of expected passengers exceeds the seat capacity. Alternatively, this is also known as the seat-shortage kilometres. Recall that P_t represents the set of allowed compositions for an arbitrary trip t. Then, let $X_{t,p}$ be a binary decision variable, where $X_{t,p} = 1$ if composition $p \in P_t$ is used for trip $t \in T$ and $X_{t,p} = 0$ otherwise. Lastly, $s^+_{t,p}$ expresses the expected number of seat shortages for trip t with composition p. Then, the seat-shortage kilometres can be written as:

$$SSK = \sum_{t \in T} \sum_{p \in P_t} \ell_t\ s^+_{t,p}\ X_{t,p}. \tag{3}$$

Lastly, SHM resembles the number of shunting movements. A shunting movement is either a coupling movement to an existing composition or a decoupling movement from an initial composition. Therefore, SHM simply represents the summation of all coupling and decoupling movements. Mathematically, SHM can be expressed as:

$$SHM = \sum_{t \in T} \sum_{\substack{(p,p') \in \Gamma_t: \\ \beta(p) \neq \beta(p')}} Z_{t,p,p'}. \tag{4}$$

Recall that $(p, p') \in \Gamma_t$ represents a feasible composition change between trip t and $v(t)$. The composition change can only represent a coupling movement, a decoupling movement or no change at all. $\beta(p)$ denotes the number of carriages of composition p and $\beta(p) \neq \beta(p')$ states that the number of carriages of the successor trip is not equal to the number of carriages of the current trip t. This indicates that a train unit is either coupled to or decoupled from the initial composition of trip t, because it is impossible to decouple and couple simultaneously. Putting this all together, the overall objective function of Equation (1) can be rewritten using Equations (2) - (4) to:

$$\begin{aligned} \min F(X, Z, N) = C_{CKM} \sum_{t \in T} \sum_{m \in M} \ell_t \; c_m \; N_{t,m} + C_{SSK} \sum_{t \in T} \sum_{p \in P_t} \ell_t \; s^+_{t,p} \; X_{t,p} \\ + C_{SHM} \sum_{t \in T} \sum_{\substack{(p,p') \in \Gamma_t: \\ \beta(p) \neq \beta(p')}} Z_{t,p,p'}. \end{aligned} \tag{5}$$

Notice that the each component of the objective function contains the summation over all trips $\sum_{t \in T}$. Therefore, it is also possible to break the objective down into an objective function which expresses the costs of a single trip. Let $t \in T$ be an arbitrary trip and let $y_t \in P_t$ denote the selected composition of this trip. Then, the objective function of an arbitrarily trip t becomes:

$$\begin{aligned} F(t, y_t, z_{t,y_t,y_{v(t)}}) = C_{CKM} \sum_{m \in M} \ell_t \; c_m \; n_{m,y_t} + C_{SSK} \; \ell_t \; s^+_{t,y_t} \\ + C_{SHM} \; z_{t,y_t,y_{v(t)}}, \end{aligned} \tag{6}$$

where $z_{t,y_t,y_{v(t)}} = 1$ if a shunting movement is necessary for the transition from composition y_t to composition $y_{v(t)}$ and $z_{t,y_t,y_{v(t)}} = 0$ otherwise. This equation can be used to compare the costs of different compositions for a trip. For Equation (5) as well as Equation (6), the parameters C_{CKM}, C_{SSK} and C_{SHM} are fundamental to determine the quality of a rolling stock schedule. These parameters should serve as the preferred trade-off between costs, service and flexibility or robustness.

4.3 Representing a Feasible Solution

If the composition model finds a feasible schedule, the values of the decision variables will be returned. However, to gain understanding which trips a specific train unit will execute, another step is required. The solution of the composition model can be transformed into a more practical schedule, which shows the exact sequence of trips a certain train unit needs to execute. This can be done using the task model, which is introduced in Nielsen (2011). This solution approach determines the sequence of trips for each train unit. The paper uses multiple solution techniques to come up with such a roster. For more information about the methods, we refer to Nielsen (2011). In this thesis, we only introduce the relevant output of the task model where any prior knowledge about the task model is not required.

Our point of interest is the exact sequence of trips for every available train unit. A rolling stock duty represents such a sequence of trips for a train unit. Let $r \in R$ be a rolling stock duty of the set of all rolling stock duties. The set R can be partitioned into subsets based on the train unit type $R_m \subseteq R$. Note that $\bigcup_m R_m = R$ and the number of rolling stock duties of type m is equal to the number of available units: $|R_m| = n_m$.

A rolling stock duty r denotes a sequence of tasks for a train unit, where a task represents a trip and the position of that specific train unit within its corresponding composition. Let K_r denote the set of tasks of a rolling stock duty $r \in R$. The sequence of tasks can be written as $K_r = \{k(1), k(2), ..., k(|K_r|)\}$. Each task k_i represent a pair $(t_{k(i)}, q_{k(i)})$, where $t_{k(i)}$ represents the ith trip of rolling stock duty and $q_{k(i)}$ denotes the position within the selected composition $p \in P_{t_{k(i)}}$. Each rolling stock duty is selected in a way such that the sequence of tasks does not violate any restrictions and is therefore feasible. For example, the succeeding trip always departs from the same station as it arrived: $s_d(t_{k(i+i)}) = s_a(t_{k(i)}),\ i = \{1, 2, ..., |K_r| - 1\}$.

When we consider unbalances between two consecutive days, the last task of the first day and the first task of the next day are of interest. Introduce the sets R^1 and R^2, where R^1 represents the set of rolling stock duties of the first day and R^2 the duties of the next day. To avoid some inconvenient notation, let t_r^0 and t_r^∞ denote respectively the first and last trip of a given rolling stock duty $r \in R$. Then, the first departure and the last arrival station of a rolling stock duty can be rewritten as respectively $s_d(t_r^0)$ and $s_a(t_r^\infty)$. Similarly, the respective departure and arrival times will be denoted as $\tau_d(t_r^0)$ and $\tau_a(t_r^\infty)$. Although the majority of the trips are carried out on every (week)day with the identical departure and arrival times, the timetable differs slightly for each day. Therefore, let us distinguish between T^1, which describes the trips of the first day, and let T^2 denote the trips of the succeeding day.

4.4 Rolling Stock Rebalancing Problem

In general, the end-of-day inventory does not equal the start-of-day inventory of the following day. Let $I_{s,m}^\infty$ be the number of train units of type $m \in M$ ending at station $s \in S$. Similarly, let $I_{s,m}^0$ be the number of train units of type $m \in M$ at station $s \in S$, which start the succeeding day. In this thesis, we only consider two consecutive days unless stated otherwise. Therefore, $I_{s,m}^\infty$ always corresponds to the end-of-day inventory of the first day and $I_{s,m}^0$ reveals the start-of-day inventory of the succeeding day.

To define an (un)balance, introduce the parameter $b_{s,m} = I_{s,m}^\infty - I_{s,m}^0$. The value of $b_{s,m}$ depicts the number of train units of type $m \in M$ in excess (or in shortage) for station $s \in S$. Note that whenever $b_{s,m} = 0$, there is a balance in train units at that station for that specific type. We assume that the number of available train units of a certain type n_m does not differ between days. At first sight, it seems obvious that the total number of surpluses is equal to the number of shortages for a certain type m. Mathematically, this can be expressed as $\sum_{s \in S} b_{s,m} = 0$ for any rolling stock type $m \in M$. Certain stations have surplus of that type ($b_{s,m} > 0$), while other stations could have a shortage of that specific type ($b_{s,m} < 0$).

Let s be a station in the set S_m^+, when there is at least one unit in surplus of type m. Equivalently, the set S_m^- corresponds to the set of stations which have a shortage of type m. Resolving one surplus at a certain station always leads to either a surplus somewhere else or a decline at another station in deficits. However, this is not entirely true. Although train units cannot magically disappear, there is an

exception. For example, if a certain train unit carries out a sequence of trips on a particular day, but has no designated trips on the succeeding day or vice versa. Here, the specified station $s_1 \in S_m^+$ is known, but there is no predefined station $s_2 \in S$ in the set S_m^-. In this case, the station s_2 can be chosen based on preferences, such as a preferred maintenance location or a station where the unit can be stored. An obvious solution would be to let the train unit stay at its last destination station s_1.

To summarise, if $\sum_{s \in S} b_{s,m} > 0$, then the number of surpluses is higher than the number of shortages of a certain rolling stock type m. Here, $\left|\sum_{s \in S} b_{s,m}\right|$ train units are not used for the following day and their designated starting station $s_d(t_r^0), r \in R^2$ is undefined. Similarly, if $\sum_{s \in S} b_{s,m} < 0$, the number of shortages is higher than the number of surpluses and some units are unused for the first day. In conclusion, although the number of available train units n_m is assumed to be equal over all days, the number of actually used train units can differ between consecutive days. The total number of surpluses is therefore not always equal to the total number of shortages and, accordingly, some units can either remain at the current station or traverse to a preferred location (e.g. maintenance site).

Let $r \in R_m$ be an arbitrarily rolling stock duty belonging to the first day of train unit type m, which ends at station $s_a(t_r^\infty)$ at time $\tau_a(t_r^\infty)$. Then, this rolling stock duty belongs to the set $B_{s,m}^1$, which is the set of rolling stock duties of the first day ending at station $s \in S$ with train unit type $m \in M$. Similarly, $B_{s,m}^2$ describes the set of rolling stock duties of the following day, which start at station $s \in S$ with type $m \in M$. Notice that $B_{s,m}^1$ and $B_{s,m}^2$ are slightly different defined: $B_{s,m}^1$ is based on the station of the last arrival $s_a(t_r^\infty), \forall\, r \in R^1$ and $B_{s,m}^2$ depends on the station of the first departing trip $s_d(t_r^0), \forall\, r \in R^2$.

The most straightforward approach of solving an unbalance is by performing a deadheading trip from one station $s_1 \in S_m^+$ to $s_2 \in S_m^-$. In this thesis, we assume that a deadheading trip is always possible, provided that this trip is executed at night. Due to the less dense usage of the rail network at night, the assumption seems reasonable and practically implementable. Moreover, we assume that a deadheading trip can depart and arrive at any station. A deadheading movement always uses the shortest possible route between two stations. The shortest route can for instance be determined by the Floyd-Warshall algorithm (Floyd, 1962). However, executing a deadheading trip could be expensive compared to the cost of a 'regular' trip. The cost of a deadheading trip is computed in a similar fashion as explained in Section 4.2. The only cost driver of an empty movement is the number of carriage kilometres. Since a deadheading trip does not have any passengers, seat-shortage kilometres are not relevant. Furthermore, we assume that no shunting movements are necessary to execute a deadheading trip or that these costs are negligible compared to the main cost driver.

The deadheading trips will be executed during the night. Because manpower is needed to perform such a trip, the costs for the staff will generally be higher than during daytime. Therefore, let us introduce a penalty factor γ for every deadheading trip. Then, the objective function for a deadheading trip with train unit type m becomes:

$$C_{s_1,s_2,m}^d = \gamma\,(\, C_{CKM}\; \ell_{s_1,s_2}\; c_m) \tag{7}$$

Here, ℓ_{s_1,s_2} denotes the distance in kilometres between stations s_1 and s_2. We assume that the extra costs of a deadheading trip are linear to the distance between the two stations. However, if one prefers to introduce a fixed cost to a deadheading movement, this could easily be altered in the objective function. For the remainder of the thesis, the costs for an empty movement from one station s_1 to another station s_2 of train unit type m will be presented as $C_{s_1,s_2,m}^d$.

5 Sequential Solution Approach

The principal disadvantage of using the composition model of (Fioole et al., 2006) is that the computation time increases rapidly when the considered time horizon extends. Especially for larger instances with many trips, the running time seems to explode when the number of selected days increases. Nevertheless, the composition model provides decent results within a reasonable time when only a single day is considered. Provided the optimality gap is set to 0%, this exact solution technique returns the optimal solution. It calls into question whether the precision of the relatively smaller instances can be used to generate rolling stock schedules for multiple consecutive days. In this chapter, we discuss how the advantages of the composition model can be re-used to solve the rolling stock rebalancing problem.

The so-called sequential solution approach splits the large problem of determining the schedule of consecutive days into smaller subproblems. In general, smaller problems are easier to solve in terms of computation time in comparison to a larger, more difficult problem. This is exactly what this solution approach does. First, the schedules of two consecutive days are calculated independently from each other and for this reason, the end-of-day inventory of the first day is in general not equal to the start-of-day inventory of the succeeding day. For the sequential solution approach, we introduce a new subproblem, which (re-)optimises a specified, narrowed time window. Given the two calculated rolling stock schedules of each day, the new subproblem attempts to dissolve the arisen unbalances. Here, all trips within the predefined time window are unfixed. To summarise, the sequential solution approach for two consecutive days can be split into three subproblems. First, the schedule of the first day is calculated and afterwards, the rolling stock roster of the second day. Then, the third subproblem (re-)optimises the initial schedules between a narrowed time window.

Let the time window of the so-called third subproblem be defined as the interval between two chosen moments in time $[\tau_1, \tau_2]$, where $\tau_1 < \tau_2$. Then, all trips $t \in T^1 \cup T^2$, for which it holds that $\tau_d(t) < \tau_1$ or $\tau_d(t) > \tau_2$, are considered to be fixed. In other words, the rolling stock schedule is recomputed for a smaller set of trips $T_P \subseteq T^1 \cup T^2$, where a trip t is included in this set if $\tau_1 \leq \tau_d(t) \leq \tau_2$. In order to resolve the unbalances, the interval must be chosen in a way such that the unbalances can actually be dissolved. In other words, time τ_1 is a certain time on the first day and τ_2 must be chosen such that it belongs to the second day. Mathematically, the interval must be chosen between $\tau_1 < \max_{t \in T^1}\{\tau_d(t)\}$ and $\tau_2 > \min_{t \in T^2}\{\tau_d(t)\}$. If one does not adhere to this condition, i.e. by setting τ_1 and τ_2 on the same day, the unbalances cannot be solved. Notice that deadheading movements, that are executed during night, are also included in the set of considered, unfixed trips T_P.

With the constructed time window, the solution approach can be explained in more detail. Firstly, use the composition model of Fioole et al. (2006) to attain the optimal rolling stock schedule of the first day, or a feasible schedule with an acceptable solution gap. Equivalently, we compute the rolling stock schedule of the second, succeeding day. Note that this schedule is independently calculated from the preceding day. Thirdly, the preferred time window $[\tau_1, \tau_2]$ must be determined. A trip $t \in T^1$, whose departure time $\tau_d(t)$ does not fit within the defined time window, will be fixed to its initial calculated composition $X_t \in P_t$ of the first step. Similarly, if it holds for any trip $t \in T^2$ that $\tau_2 < \tau_d(t)$, then the composition of this trip is fixed. In the last step, the composition model re-used to compute the rolling stock schedule of the specified time window. The eventual output is a schedule of two days where the end-of-day inventory is equal to the start-of-day inventory of the following day.

Technically, all trips $T^1 \cup T^2$ are used as input for the third subproblem, where only the trips within set T_P are unfixed. Instead of fixing the set of trips, one could also remove the fixed trips from the set $T^1 \cup T^2$ to reduce the number of decision variables. However, simply removing the fixed trips from the input is not an option, because trips are connected to each other by their predefined composition changes. Extracting trips from the set $T^1 \cup T^2$ would lead to conflicts to the specified transitions and composition changes.

The choice of the interval $[\tau_1, \tau_2]$ is the most essential decision of this solution approach. The interval specifies the chosen set of trips to be recomputed. If the width of the time windows expands, the number of selected trips increases and the number of fixed trips decreases. Moreover, in general, the (expected) running time of the third subproblem increases. In order to determine an acceptable time window, knowledge about the railway system is necessary. For example, if the first train on the second day departs at 05:15, then one would be indifferent for a choice between [23:00, 04:00] or [23:00, 05:00]. Both time windows have the exact same set of considered trips T_P. Note that the choice of whether a trip is included in the set of unfixed trips T_P depends on the departure time. Contrarily, considering the arrival time $\tau_a(t)$ instead of the departure time will lead to similar, yet slightly different, results.

In general, there are more shunting possibilities aside from the rush hours. Therefore, a starting point could be to recompute the rolling stock schedule after the evening rush hours of the first day and before the morning rush hours of the next day (i.e. [19:00, 06:00]). If one finds the retrieved solution value to be unsatisfactory, the time window could be expanded even further, at the cost of an expected, enlarged running time. If the time window is expanded as much as possible [00:00, 23:59], the third subproblem is just as difficult as the initial, larger problem. In conclusion, there is a trade-off between the quality of the objective function and the running time.

6 Heuristic Approach

In this chapter, a more sophisticated method will be explained. Again, the goal is to dissolve all unbalances such that the overall schedule is in balance. First of all, Section 6.1 provides a (linear) mathematical formulation which serves as the upper bound for the rolling stock rebalancing problem. This upper bound can be used to determine the quality of the proposed solution approaches of this chapter. In Section 6.2, we address a method which solves only one unbalance. It uses a time-dependent graph to represent all possibilities of a train unit to manoeuvre. The shortest path is computed using topological sorting and is used to describe a feasible solution to dissolve one unbalance. In general, the number of unbalances is higher than one and Section 6.3 describes how multiple unbalances can be solved using the shortest paths of time-dependent graphs. This path generation approach can also be seen as a self-contained solution approach for the rolling stock rebalancing problem, since it dissolves all unbalances. Then, Section 6.4 suggests multiple neighbourhoods that tend to generate a different end-of-day and/or start-of-day inventory. By using simulated annealing, the neighbourhoods randomly search for changes in the inventory levels. By changing the inventory level, more or less unbalances could arise. In combination with the upper bound or the path generation approach, simulated annealing can also dissolve all unbalances and, for this reason, can also be regarded as a solution approach for the rolling stock circulation problem. Lastly, Section 6.5 presents an overall solution approach which iteratively uses the methods of Section 6.3 and 6.4. In total, this chapter provides three different solution approaches for the rolling stock rebalancing problem.

6.1 Calculating the Lower and Upper Bound

In this section, we describe how the lower and upper bound of the rolling stock rebalancing problem can be defined. First of all, let us discuss how the lower bound can be determined. Using the composition model, the rolling stock schedule of a single day can be calculated. Let $F(x_i)$ denote the optimal solution value of the rolling stock schedule of day i. Then, the lower bound can be written as: $LB = F(x_1) + F(x_2)$. The lower bound is simply the summation of the solution values of two consecutive days. Here, the optimal solution value is equal to the lower bound if there are exactly zero unbalances between the schedules x_1 and x_2. Otherwise, the cost of dissolving the remaining unbalances is positive and, therefore, the optimal solution value is strictly higher than the lower bound.

The upper bound is calculated using a linear program which solves a given set of unbalances between two consecutive days. This is done by assigning a deadheading trip to each unbalance. As input, similarly to Chapter 5, we take the (near-)optimal rolling stock schedules of each individual day. Because the cost of a deadheading trip serves as an upper bound for one unbalance, we only need to find the best way to dissolve all unbalances.

The upcoming linear program returns the least-cost option to dissolve all unbalances while only using deadheading trips. Rolling stock types are independent of each other when deadheading trips are considered. By way of explanation, a shortage of type $m_1 \in M$ can never be dissolved by transporting a train unit of type $m_2 \in M$ to that station. Furthermore, the costs of empty movements are independent of different types of train units. Using multiple train units of a different type for a single deadheading trip does not assumably lead to a (significant) cost reduction. Therefore, the linear program is expressed for an arbitrary train unit type $m \in M$. In order to obtain the actual upper bound, the linear program needs to be solved for every type $m \in M$.

In this linear program, our goal is to minimise the costs of the deadheading trips. The cost of an empty movement from a certain station $s_1 \in S_m^+$ to another station $s_2 \in S_m^-$ with type $m \in M$ equals $C^d_{s_1,s_2,m}$. Recall from Equation (7) that the cost of an empty movement between two stations depends on two factors: the travelled distance and the number of carriages. Then, let x_{s_1,s_2} be the integer decision variable which is equal to the number of deadheading trips from station $s_1 \in S_m^+$ to $s_2 \in S_m^-$. To decrease the total number of decision variables, x_{s_1,s_2} can be defined for the compatible cases. It is only relevant to evaluate deadheading trips between stations in excess and stations in shortage of a certain type m. In an optimal solution, deadheading trips will never start at a station in shortage and, similarly, an empty movement will never end at a station in excess. Therefore, x_{s_1,s_2} is defined for every station $s_1 \in S_m^+$, where $b_{s_1,m} > 0$ and every $s_2 \in S_m^-$ for which $b_{s_2,m} < 0$. Putting this all together, leads to the following linear program to dissolve all unbalances of rolling stock type $m \in M$:

$$\min \quad \sum_{s_1 \in S_m^+} \sum_{s_2 \in S_m^-} C^d_{s_1,s_2,m}\, x_{s_1,s_2} \tag{8}$$

$$\text{s.t.} \quad \sum_{s_1 \in S_m^+} \sum_{s_2 \in S_m^-} x_{s_1,s_2} = \min \left\{ \sum_{s_1 \in S_m^+} b_{s_1,m}, - \sum_{s_2 \in S_m^-} b_{s_2,m} \right\}, \tag{9}$$

$$\sum_{s_2 \in S_m^-} x_{s_1,s_2} \leq b_{s_1,m}, \qquad \forall s_1 \in S_m^+, \tag{10}$$

$$\sum_{s_1 \in S_m^+} x_{s_1,s_2} \leq -b_{s_2,m}, \qquad \forall s_2 \in S_m^-, \tag{11}$$

$$x_{s_1,s_2} \in \mathbb{Z}^+, \qquad \forall s_1 \in S_m^+, s_2 \in S_m^-. \tag{12}$$

The objective function (8) minimises the costs of the selected empty movements. Constraint (9) ensures that every unbalance is solved. As described in Section 4.4, the number of actual used train units can differ between days and the right side of the constraint represents the minimum number of unbalances that are enforced to be dissolved. Constraint (10) makes sure that deadheading trips only leave at stations with a surplus. Moreover, it guarantees that the number of departing deadheading trips does not exceed the number of train units in surplus. Similarly, Constraint (11) makes sure that deadheading trips arrive at a station with a deficit. Lastly, Constraint (12) ensures that the decision variable x_{s_1,s_2} is an integer decision variable.

6.2 Solving One Unbalance

In this section, we consider only a single unbalance between two stations. The goal is to transfer this train unit from the station in excess to its designated station in shortage with the least amount of costs. The proposed solution approach constructs a time-dependent graph where every possibility to traverse a train unit is represented by nodes and edges. First, Section 6.2.1 introduces the problem and provides an overview of the solution approach. Furthermore, this section demonstrates when nodes and edges are added to the graph. Secondly, Section 6.2.2 explains how the weight of an edge is calculated. Lastly, Section 6.2.3 describes in the order nodes and edges are added to the graph. This order is important, because, the shortest path is computed using topological sorting. Furthermore, this section discusses how and when nodes are removed from the graph. Removing redundant nodes could speed up the process of finding the shortest path. Lastly, Section 6.2.4 discusses how a time-dependent graph can be re-used when the number of considered unbalances between two stations increases.

6.2.1 Time-Dependent Graph

In this section, the idea of the time-dependent graph will be outlined. First, the definition of the graph and its corresponding nodes and edges will be declared. Eventually, the time-dependent graph should represent all possibilities for a certain train unit to traverse from a station in surplus to the station with a shortage. This section describes how the nodes and edges represent the possibilities and when nodes and edges are included in the graph. A few examples are used to illustrate the concept of the time-dependent graph.

First of all, let us start with the definition of a time-dependent graph. Let $G = (V, E)$ represent a directed, weighted graph where V and E denote respectively the set of vertices and the set of edges. A node $v \in V$ represents a station $s_v \in S$ and, because this graph is time-dependent, it also indicates a certain moment in time τ_v. Therefore, a node can also be denoted as a pair: a station and a specific moment in time (s_v, τ_v). Furthermore, let $e \in E$ represent a directed edge from a node $v_1 \in V$ to another node $v_2 \in V$. By definition, an edge cannot have the same node as a tail v_1 and a head v_2, and duplicate edges do not exist. The set of edges can be written as $S = \{(v_1, v_2) \mid (v_1, v_2) \in V^2, v_1 \neq v_2\}$. For any arbitrary edge (v_1, v_2), it must hold that $\tau_{v_1} < \tau_{v_2}$. Consequently, (negative) loops do not exist in a time-dependent graph G.

Initially, our goal is to find the shortest path from a source node to a destination node. Let $V_l \subset V$ represent the set of source nodes and let $V_f \subset V$ describe the set of destination points. At this moment, both sets have only one element, but in Section 6.2.4, these sets will be expanded. The intention is to find the shortest path from any element of V_l to any arbitrary node in V_f. Let us clarify the source and destination nodes with an example. Suppose that the last trip of a train unit of type m arrives at its last designated station $s_a(t) = s_f$, where $s_f \in S_m^+$. This station has one unit in excess and an arbitrary station $s_l \in S_m^+$ has one unit of type m in shortage. To be more precise, the last trip of a train unit ends at station s_l at time τ_l and the goal is to bring that specific unit as cheap as possible to station s_f before time τ_l, such that the train unit can execute the trips of the following day. In other words, we want to find the cheapest path from node $v_l \in V_l$ (s_{v_l}, τ_{v_l}) to node $v_f \in V_f$ (s_{v_f}, τ_{v_f}).

Lastly, before getting into more details, there are a few assumptions for the time-dependent graph. First of all, as already mentioned, deadheading movements at night are always possible. After the last departed trip at station s_l, a train unit can execute an empty movement from station s_l to any other station s_f. The costs of a deadheading trip from s_l to s_f is considered to be an upper bound for the shortest path between v_l and v_f. Furthermore, we assume that stations have unlimited space for temporary unused train units. Later in this section, we will discuss how one can deviate from this assumption.

As mentioned before, the nodes and edges of the time-dependent graph represent all possible paths a train unit of type $m \in M$ can traverse between two stations. The time-dependent graph is different for each rolling stock type. Besides having distinct costs, the shunting possibilities and the feasible composition of a trip are different for each type. Obviously, a different starting (s_l, τ_l) and/or ending point (s_f, τ_f) leads to a different approach and, therefore, a new graph is also necessary. The input for the time-dependent graph is a starting point (s_l, τ_l), an ending point (s_f, τ_f) and a rolling stock type $m \in M$. In conclusion, each triplet (s_l, s_f, m) requires a unique time-dependent graph, because the shortest path between s_l and s_f is different for every rolling stock type $m \in M$.

In short, each (shunting) possibility that is feasible for a certain rolling stock type m will be included in the time-dependent graph. To clarify the time-dependent graph in words, a node represents a location and a moment in time. An edge is drawn between two nodes $v_1, v_2 \in V$ if the given train unit can travel from (s_{v_1}, τ_{v_1}) to (s_{v_2}, τ_{v_2}) without violating any imposed constraints. Here, the selected rolling stock type can only be added to existing compositions. Changing or even removing some train units from the initial composition is not considered as a possibility. These prohibited changes alter the composition of the succeeding trips and, eventually, these changes could lead to an adjusted end-of-day and/or start-of-day inventory. This could lead to even more (or less) unbalances, while we are simultaneously solving an unbalance. Therefore, only the considered train unit can execute any additional shunting movements. Let us briefly outline the various cases when an edge can be included:

- A train unit is currently stored at a station and its re-allocation time has passed. Then, for the next leaving trip:
  - the train unit can couple to this trip, if possible;
  - remains at the current station.
- A train unit is coupled to a trip and it arrives at a station:
  - Decouple from the current composition, if possible;
  - Continue its journey with the successor trip, if possible.
- If a train unit is decoupled from a trip:
  - Wait at the station until its re-allocation time has passed.

Each of the above possibilities is labelled as a relevant event. For every relevant event, a node is added to the time-dependent graph. The time at which the event occurs and its location are stored in the node. Then, an edge is added between this new node and the node corresponding to the previous, relevant event. The following examples will clarify in more detail when nodes and edges are added to the graph.

Example 1 First of all, let us start with a simplified example. The starting point is a train unit of type m whose last trip of the first day arrives at station $s_l = A$ at $\tau_l = 22{:}00$. The next day, the train needs to be at station $s_f = B$ at $\tau_f = 05{:}00$. An example of the corresponding time-dependent graph for this example can be seen in Figure 1. The source and the destination node are illustrated as white circles. Let us start with an important remark. Although the train unit is physically present at station A at $\tau_l = 22{:}00$, it is (usually) impossible to couple the unit to another trip, which leaves just a few minutes after its arrival at 22:00. A train unit has a re-allocation time $\varrho(s)$ that needs to be respected before it can perform any shunting movements. Let us assume that the re-allocation time is equal to 30 minutes.

The re-allocation time is also visible in Figure 1. Node $(A, 22{:}30)$ represents the time at which the train unit is available for any coupling movements. Instead of adding both a source node and a node for the re-allocation time, one can also choose to only add node $(A, 22{:}30)$ to the graph. After 22:30, four trips depart from station A at respectively departure times 22:35, 22:45, 23:05 and 23:35. These trips are also enumerated in Figure 1. Notice that the trip departing at 23:05 is not

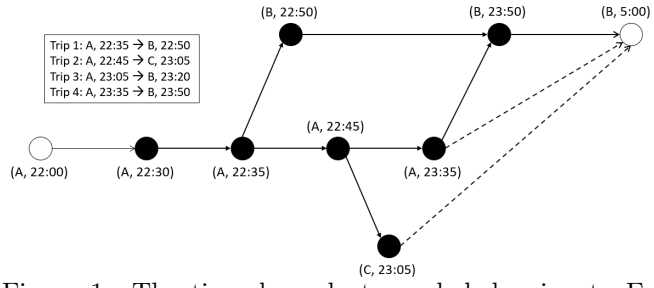


Figure 1: The time-dependent graph belonging to Example 1. Here, the basics of the graph are explained.

visible in the graph. We will discuss the departing trips in a chronological order and eventually explain why there is not a node corresponding to trip 3. First of all, let us consider the trip departing at 22:35. To determine whether this node is a relevant event, it is necessary to check whether train unit m can be coupled to the composition of trip 1. As a matter of fact, there is one feasible composition for trip 1. Therefore, node $(A, 22{:}35)$ is added to the graph and an edge between node $(A, 22{:}30)$ and $(A, 22{:}35)$ is included.

Considering the newly created node, the train unit has two options: it can be coupled to trip 1 or remain at station A. Therefore, two edges are added to the graph: one which represents staying at station A and one which illustrates a coupling movement and executing trip 1. The edge, which represents staying at the current station, is connected to the next event. Here, the next event is the following trip where train unit m can be coupled to. We check and confirm that the train unit can be coupled to trip 2 and therefore the node $(A, 22{:}45)$ and its connecting edge are added to the graph. Similarly, we check whether the train unit can be coupled to the trips departing at 23:05 and 23:35. The train unit cannot be coupled to trip 3 and, therefore, this node will not be included in the graph. However, the train unit can be coupled to trip 4 and its corresponding node and edge are added to the graph. If the assumption of unlimited storage space at a station is dismissed, more caution is required for adding edges between nodes representing the same station. For example, if the maximum capacity at station A has been reached between 22:55 and 23:05, then the edge between node $(A, 22{:}45)$ and $(A, 23{:}35)$ is invalid and consequently non-existing for this time-dependent graph.

Now, let us consider the cases where the train unit is coupled to a trip. The destination of the first trip is station B at 22:50. Here, the train unit has only one option and, that is, staying at station B. Therefore, node $(B, 22{:}50)$ and its connecting edge to node $(A, 22{:}35)$ are added to the graph. Similarly, the second trip arrives at final destination C at 23:05 and node $(C, 23{:}05)$ is added to the graph. Lastly, the trip departing at 23:35 arrives at final destination B at 23:50. Besides adding node $(B, 23{:}50)$ and its corresponding edge, an edge between node $(B, 22{:}50)$ and $(B, 23{:}50)$ is also included in the graph. This is due to the reason that the train unit is in storage at station B from 22:50 and is still in storage at 23:50.

The dashed lines represent the empty movements, which are drawn between all accessible stations in the evening and all accessible stations in the following morning. A station is so-called accessible in the evening if the train unit can be decoupled from at least one trip arriving at this station. Similarly, a station is accessible in the morning if there exists a path starting from that station and arrives in time at station s_f before its departure time τ_f. Assume that the destination node represents the first departing train at station B and no trips arrive at station B before 05:00. In other words, the only accessible station of the following morning is station B, because no trip arrives before 05:00 at station B. Therefore, only dashed lines to station B are drawn. Notice that the edge between $(B, 23{:}50)$ and $(B, 05{:}00)$ is technically not an empty movement, because the train unit is already at station B.

Example 2 The previous example gave a more clear illustration and explanation of the time-dependent graph. In the upcoming examples, the general idea of the graph is assumed to be understood. Hence, we exploit the special structure of graphs due to slightly unusual situations. In example 2, visible in Figure 2, two particular events are described in more detail: the re-allocation time and the introduction of a trip after midnight. In this time-dependent graph, the starting node $(A, 22{:}30)$ is the time at which the re-allocation time has passed.

Three relevant events happen at station B before midnight: two arrivals at 22:50 and 23:20, and a departure at 23:25. The train unit can couple to trip 1 as well as to trip 2. These nodes and related edges are added to the graph. Trip 3, leaving at 23:25, is worth considering thoroughly. If the train unit is present at station B, then the train unit can be coupled to this trip. However, due to the re-allocation time ($\varrho(B) = 30$), only a decoupled train unit of trip 1 can eventually be coupled to trip 3. If one selects to couple to trip 2, then the train unit cannot be coupled to trip 3. Therefore, an edge between node $(B, 23{:}20)$ and node $(B, 23{:}25)$ is not drawn.

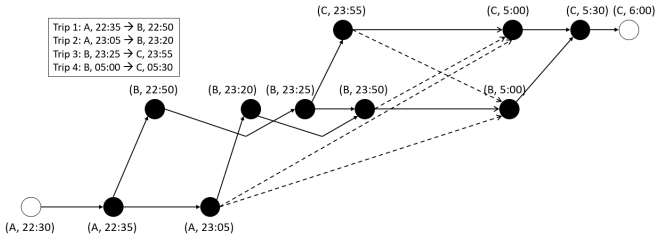


Figure 2: The time-dependent graph belonging to Example 2. Here, the focus lies on the re-allocation time and the edges representing empty movements.

Secondly, note that there is a trip on the following morning, which arrives at our preferred end station C before 06:00. Similarly to the trips in the evening, a node and an edge are included in the graph, if the train unit can be coupled to the trip leaving at 05:00. Apparently, this is correct and an edge connects node $(B, 05{:}00)$ and the destination node $(C, 06{:}00)$. Therefore, station B is also an accessible station in the morning, because there is a trip which reaches the destination node before the designated time τ_f. Observe that the number of edges representing a deadheading trip increases. The assumption is that a deadheading trip is included between any accessible set of stations. Although a deadheading movement from station C to station B is highly unlikely to ultimately belong to the shortest path of this graph, the edge is still included.

Example 3 In the third example, we examine how the graph is drawn when some composition changes or decoupling movements are infeasible. In Figure 3, three trips are drawn which all depart from station A, via B and arrive at station C. The selected train unit can be coupled to every trip and, therefore, the corresponding nodes are included in the time-dependent graph. Moving on to the next station B, the train unit can be decoupled from the composition of the first and the third trip. However, for some reason, the train unit cannot be decoupled from or coupled to the second trip. Nonetheless, the train unit may be decoupled at station B during the first trip and coupled to the third trip, even regarding the re-allocation time. Therefore, an edge between these two nodes is drawn. Lastly, the first trip cannot decouple at station C, but the remaining trips can decouple at station C. Therefore, the first trip eventually leads to a dead-end. If there were no limitations to the (de)coupling movements, the number of possible paths between the source and the destination node equals ten. But, due to the shunting limitations, this number reduces to six feasible paths.

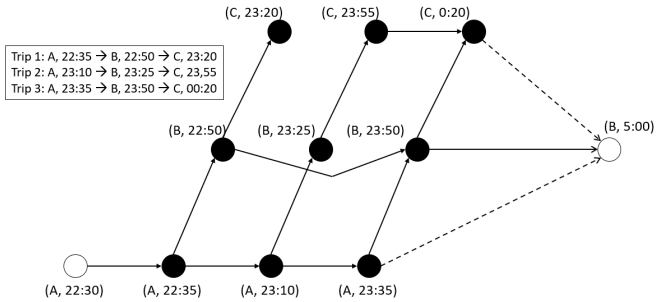


Figure 3: The time-dependent graph belonging to Example 3. Here, the focus lies on restricted de(coupling) movements, which limits the number of feasible paths.

Example 4 Finally, let us consider a special case where the position of the selected train unit leads to a different path. In Figure 4, there is only one trip which leaves at station A and proceeds through stations B, C and D and subsequently travels the same route back to station A. This instance is interesting because the selected train unit can be coupled to the front as well as to the rear of the initial composition

at station A. At first sight, this does not seem like a difference, because the number of carriage kilometres and the seat-shortage kilometres increases proportionally. However, it is necessary to draw two lines and two different nodes for each possibility. Therefore, three edges are leaving the node $(A, 22{:}35)$: two edges corresponding to the two different compositions and one edge because the selected train unit could stay at station A.

At station B and C, there are no shunting movements executable and, therefore, there is only one directed edge included for each corresponding node. At station D, a train unit can only be decoupled at the rear. Therefore, the train unit, which was coupled at the rear of the initial composition, has two options: decouple at station D or continue its journey towards station C. The train unit coupled at the front has only one option, which is continuing with the successor trip towards station C. Next, consider both return journeys, where both trips leave station D and travel towards station C, then B and eventually station A. However, due to some restrictions, the chosen composition, where the selected train unit is at the rear, is infeasible for the trip from C to B. Because there are no shunting possibilities at station C, this path leads to a dead-end.

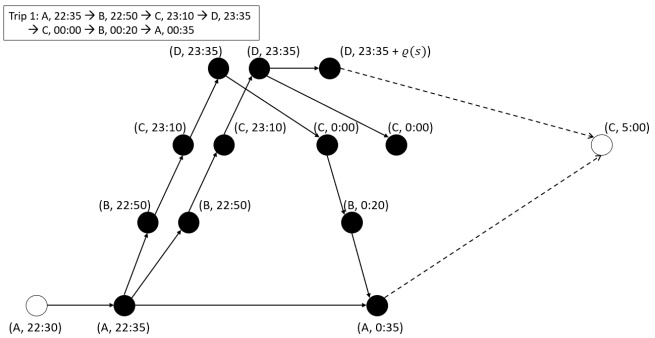


Figure 4: The time-dependent graph belonging to Example 4. Because two different coupling movements are possible for one trip, two nodes are necessary to represent the different possibilities.

In conclusion, the time-dependent graph is a combination of nodes and edges which visualise all possibilities a selected train unit can traverse from a starting point (s_l, τ_l) to an ending node (s_f, τ_f). Edges illustrate either a feasible (de)coupling movement, represent staying at the current location or executing an empty movement. Nodes are accessible from each other if there is a directed edge between them. Dashed lines display the empty movements, which are drawn from all accessible train stations between the first day and the second day.

6.2.2 Costs of a Weighted Edge

Before the shortest path of the time-dependent graph can be calculated, we need to determine the weight of each edge. Let c_{v_1,v_2} denote the costs of an edge between the two vertices $v_1, v_2, \in V$. The costs depend on the situation an edge represents. For instance, the dashed lines of the previous time-dependent graphs illustrate a deadheading movement between two stations. Moreover, edges between nodes, which represent the identical station, do not have any costs. On the other hand, edges representing a trip have additional carriage kilometres, (possibly) seat shortages and/or costs for (de)coupling movements. Each edge belongs to either one of the following five situations:

- An edge represents "staying at the station": $c_{v_1,v_2} = 0$.
- An edge represents "coupling and executing the trip". Let X_t represent the initial composition of the trip and let X_t^* be the new composition of trip t. Furthermore, $p(t)$ denotes the predecessor trip of t. Then, by using Equation (6), the costs are $c_{v_1,v_2} = F(t, X_t^*, 1) - F(t, X_t, z_{p(t),X_{p(t)},X_t})$. Note that $z_{p(t),X_{p(t)},X_t}$ could be equal to 1, if there is already a coupling movement between trip $p(t)$ and t in the initial schedule.

- An edge represents "executing the successor trip". Here, let t be the successor trip and let $p(t)$ represent the predecessor trip of t. Similarly to the previous situation, the costs are $c_{v_1,v_2} = F(t, X_t^*, z_{p(t),X_{p(t)}^*,X_t^*}) - F(t, X_t, z_{p(t),X_{p(t)},X_t})$.
- An edge represents "decoupling from a trip": $c_{v_1,v_2} = C_{SHM}$. An example of this situation would be the edge between node $(D, 23{:}35)$ and $(D, 23{:}35 + \varrho(s))$ in Figure 4. In some of the previous graphs, the costs of decoupling is implicitly included in the node or edge, like in Figure 1.
- An edge represents "a deadheading trip" between station s_1 and s_2: $c_{v_1,v_2} = C^d_{s_1,s_2,m}$.

Technically, some edges could have negative costs. Suppose there is a trip t of the initial rolling stock schedule where the number of estimated passengers exceeds the number of seats. If one couples a train unit to the initial composition X_t, the altered composition X_t^* could possibly dissolve the shortage of seats. Decreasing the number of seat shortages decreases the objective function as well. If the decline of seat shortages exceeds the additional carriage kilometres costs, then an edge c_{v_1,v_2} has negative costs. In Section 6.2.3, we will explain why negative weighted edges will not lead to any complications.

Lastly, let d_{v_l} denote the shortest path from the source node to an arbitrarily node v_1. If the current distance of a node d_{v_1} exceeds the costs of the deadheading trip $C^d_{s_l,s_f,m}$, then a path through node v_1 never leads to the shortest path. Because the shortest path will never go through this node, the node is redundant and could be excluded from the graph. When constructing the time-dependent graph, one could take this rule into account. Moreover, one could even assemble a stricter rule, which removes even more redundant nodes. Suppose that the minimum cost from an arbitrarily node v_1 to the destination node v_f is equal to $C^L_{v_1,v_f}$. Let s_{v_1} be the corresponding station of node v_1. If it holds that $s_{v_1} = s_f$, then $c_{v_1,v_f} = 0$ and if $s_{v_1} \neq s_f$, then $c_{v_1,v_f} > 0$. For the last situation, the train unit can either perform a deadheading trip from s_{v_1} to s_f or it is coupled to one or multiple trips to eventually end at station s_f. If the train unit cannot be coupled to a trip with a seat shortage, then none of the remaining edges have negative costs. Therefore, the minimum cost from a node v_1 to a destination node v_f with train unit of type m is equal to:

$$C^L_{v_1,v_f} = \min\{C^d_{s_{v_1},s_f,m},\ C_{CKM}\ \ell_{s_{v_1},s_f}\ c_m + C_{SHM}\}, \tag{13}$$

provided none of the succeeding trips have a seat shortage. Here, $\ell_{s_{v_1},s_f}$ is the distance in kilometres from station s_{v_1} to station s_f. The minimum cost is either equal to a deadheading trip or to at least one coupling movement to a trip travelling towards station s_f. If, for any node $v_1 \in V$. it holds that $d_{v_1} + C^L_{v_1,v_f} > C^d_{s_l,s_f,m}$, then this node is redundant and could be excluded from the graph.

6.2.3 Solving the Shortest Path Problem

By eventually examining all feasible possibilities, a time-dependent graph is obtained. The shortest path of this graph represents the cheapest possibility to dissolve an unbalance between two stations. As already noted, it is theoretically possible that the edges have a negative weight. If a train unit is added to the initial composition, it generally leads to higher costs because of the increase in carriage kilometres and the number of shunting movements. However, if the benefit of dissolving an already existing seat shortage is higher than the cost of adding an additional train unit, the weight of the corresponding edge is negative. In practice, this will happen sporadically, but consequently, Dijkstra's algorithm (Dijkstra et al., 1959) cannot be used.

However, the time-dependent graph has a fundamental characteristic: the graph can never have a (negative) cycle. Because nodes are time-dependent, a node can never be visited twice for any arbitrary

path. The problem can be redefined as finding the shortest path in a directed acyclic graph (with negative weights). Accordingly, topological sorting can be used to find the shortest path in $O(E+V)$ time. Marchetti-Spaccamela et al. (1996) state for any pair of vertices $N, N' \in V$ of a directed acyclic graph $G = (V, E)$, that vertex N always precedes vertex N'. If this applies for every node of a graph, then the set of nodes is topologically sorted. In a time-dependent graph, edges are only included between two nodes $v_1, v_2 \in V$ if the corresponding time τ_{v_1} is strictly lower than τ_{v_2}. Therefore, the nodes of a time-dependent graph are already in a topological order, provided the nodes are sorted chronologically.

In Algorithm 1, an overview of the topological sorting can be seen. This method simultaneously adds nodes to the graph and computes its corresponding distance from the starting node. Furthermore, it checks whether the node could possibly be removed from the graph, because a path through this node leads to a higher cost than the predefined upper bound. In Lines 1-6, the algorithm is initialised. The graph $G = (V, E)$ is defined as an empty set of vertices and edges, except the included set of source nodes V_l. The distance of each source node is set to $d_v = 0, \forall\, v \in V_l$.

Then, the algorithm continues to search for new nodes until no more nodes can be added to the graph. We need to find the first relevant event after time τ, which corresponds to the time of node N. As already explained, a relevant event is any occasion where a node is added to the graph. For example, if the first relevant time after τ is a departure from station s_l, but due to a restriction, it is not possible to couple to this trip, then this event is not relevant. In this case, this situation will not be considered and one needs to search for the next relevant node. However, if a coupling movement is feasible for this trip, then the event is represented as node N with time τ_N (Line 8 - 9). Line 10 states that the set E' represents all edges which have a directed edge from any $N' \in V$ to the new node N. Afterwards, Line 11 declares that all edges E' and the new node N are added to the current graph $G = (V, E)$.

Algorithm 1: Topological Sorting to find the Shortest Path

```
   Result: Shortest Path from V_l to V_f
 1 V = ∅: set of vertices;
 2 E = ∅: set of edges;
 3 G = (V, E): a directed, time-dependent graph;
 4 V_l ← source nodes, V_f ← destination nodes;
 5 N ← V_l, d_v ← 0, ∀ v ∈ V_l;
 6 G = (V, E) ← G = (V ∪ N, E);
 7 while N ≠ ∅ do
 8 |   Find the first event after time τ;
 9 |   Define the new node N with corresponding time τ;
10 |   Determine all incoming edges E' of N;
11 |   Update G = (V, E) ← G = (V ∪ N, E ∪ E');
12 |   d_N ← ∞;
13 |   for every adjacent node N' of N do
14 |   |   if d_N > d_N' + c_N',N then
15 |   |   |   d_N = d_N' + c_N',N;
16 |   |   end
17 |   end
18 |   if C^d_{s_l,s_f,m} < d_N + C^L_{N,N_E} then
19 |   |   Remove N and its corresponding edges E' from the graph G = (V, E);
20 |   end
21 end
```

Lines 12 till 17 represent how the distance is updated using the topological sorting. Because all vertices are already topologically sorted, only the distance of the newly added node N needs to be updated. First, its distance is set to $d_N = \infty$. Then, for every incoming edge of node N, the condition $d_N > d_{N'} + c_{N',N}$ needs to be checked. Here, N' is the preceding node of node N and the cost of the directed edge from

N' to N is equal to $c_{N',N}$. Lastly, Lines 18-20 check whether the new node N is redundant for this time-dependent graph. A node is considered as redundant if either the upper bound has been reached or there does not exist a directed edge which starts at this node, a so-called dead-end.

6.2.4 Multiple Source/Destination Nodes

In the time-dependent graph, the assumption was initially that there is only one source node and one destination node. These nodes correspond to the last arrival time and location of the first day (s_l, τ_l) and the first departure time and location of the second day (s_f, τ_f) respectively. However, if multiple train units of type m end at station s_l, then each train unit has a different arrival time. In this section, we explain which arrival τ_l and departure time τ_f should be chosen for the time-dependent graph and how multiple unbalances between two stations can be dissolved.

To describe to the problem in more detail, introduce two rolling stock duties of train unit type m: $r_1, r_2 \in B^1_{s,m}$ whose last trips end at a certain $s_1 \in S^+_m$. Suppose that r_1 and r_2 are the only two train units which end at this station, but their arrival times are different: $\tau(t^\infty_{r_1}) < \tau(t^\infty_{r_2})$. Moreover, suppose that there is also a shortage of two units at a certain station $s_2 \in S^-_m$. Its corresponding rolling stock duties are $r_3, r_4 \in B^2_{s,m}$, where the departure station of the first trip is s_2. Both train units need to traverse from station s_1 to s_2. The shortest path would dissolve one unbalance, but the question rests how the second unbalance can disappear.

To give a concrete example, take a look at Figure 5. The number of source and destination nodes has increased and grey nodes are included in the graph. The first grey node in the left corner represents the 'actual' source node and connects to the rolling stock duties r_1 by node $(A, 22{:}30)$ and r_2 by node $(A, 22{:}55)$. Both nodes belong to the set V_l. Here, 22:30 and 22:55 denote the time at which the train unit of respectively duty r_1 and r_2 is available after its re-allocation time has passed. Similarly, the grey node on the right of Figure 5 connects to rolling stock duties r_3 and r_4, where $(B, 05{:}00), (B, 05{:}01) \in V_f$.

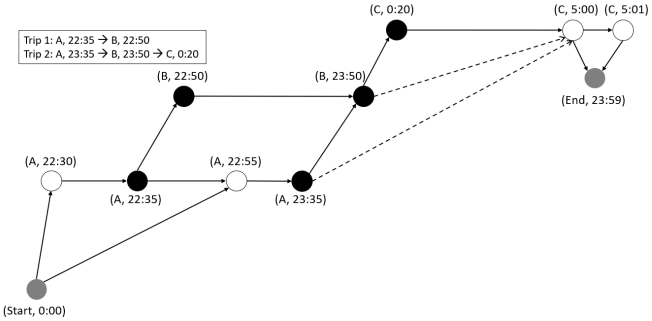


Figure 5: A time-dependent graph with two source and two destination nodes.

The edge between a grey and a white node represents a specific train unit. If a path contains this edge, it implies that this specific train unit is used to dissolve an unbalance. Therefore, if both unbalances need to be dissolved, there must be two distinctive paths, where both paths imply using a particular train unit. Considering Figure 5, suppose that the shortest path of this time-dependent graph contains the edge between the grey node and the white node $(A, 22{:}30)$. Then, the next path must contain the edge between $(\text{Start}, 00{:}00)$ and $(A, 22{:}55)$ such that both unbalances are dissolved by two unique train units. This can be done by (temporary) removing the edge between $(Source, 00{:}00)$ and $(A, 22{:}30)$ from the time-dependent graph. Then, the distance of every node needs to be recomputed and eventually, the output would be the shortest path containing node $(A, 22{:}55)$. To summarise, the unbalances are dissolved by the shortest path of the entire graph and the shortest path of the graph, where the edge between the grey node $(Source, 00{:}00)$ and $(A, 22{:}30)$ is removed.

Suppose that the two obtained paths both contain the edge between nodes $(B, 23{:}50)$ and $(C, 00{:}20)$. This means that the new composition of this trip consists of two additional train units of type m compared to the composition of the initial schedule. Theoretically, this could be possible, provided it does not lead to any violations of the predetermined composition and composition changes. However, in practice, this is rather uncommon. If two train units are added to an initial composition, the composition will contain at least three units. Especially in the evening, the majority of the seats will be unused, but more importantly, the length of the composition could possibly exceed the maximum length of a trip. Therefore, we assume that only one train unit can be coupled to the initial composition of a trip. The second unbalance can be tackled by removing the edge between nodes $(B, 23{:}50)$ and $(C, 00{:}20)$ and recomputing the distances of the vertices.

In conclusion, multiple unbalances between two stations can be solved using the time-dependent graph. It is essential to check whether one specific train unit dissolves only one unbalance and that only no more than one train unit can be coupled to a composition of the initial schedule. By temporary removing an edge of the time-dependent graph, multiple paths can be obtained such that all unbalances between the two stations are dissolved without violating any constraints.

6.3 Solving Multiple Unbalances

In this section, we explain how all arisen unbalances can be dissolved by using the shortest path of a time-dependent graph. In the previous section, topological sorting is introduced, which could solve one or multiple unbalance(s) between two stations. This method only considers one time-dependent graph of a triplet (s_l, s_f, m). Yet, in practice, multiple stations have a surplus or a shortage of a train unit. The following solution approach, also referred to as path generation, can solve all unbalances by considering multiple time-dependent graphs. Here, a solution approach is described which generates paths between stations in surplus and stations in shortage. The method is similar to column generation, because paths are iteratively added until a stopping criterion has been met. Ultimately, the path generation approach can dissolve all unbalances and, for this reason, is a self-contained solution approach for the rolling stock rebalancing problem.

First, Section 6.3.1 explains the master problem. This mathematical formulation tries to assign a path to each unbalance. The rolling stock rebalancing problem is solved if all unbalances are covered by a path. Subsequently, Section 6.3.2 provides a general overview of the algorithm and explains why it is similar to column generation. Furthermore, it explains in detail how the paths are generated.

6.3.1 Master Problem

Before getting into the details of the mathematical formulation, let us first describe the general idea of the algorithm. Each time-dependent graph has a corresponding triplet (s_l, s_f, m), which describes all possibilities for a train unit of type m to traverse from station s_l to station s_f. The shortest path represents a combination of adjusted compositions and/or deadheading movements. These adjustments eventually dissolve the unbalance between s_l and s_f. The idea is that paths from the time-dependent graphs are iteratively generated until all unbalances are dissolved. Here, similarly to Section 6.2.4, one should be aware that no more than one train unit can be added to the initial composition of a trip and that only one path, in a chosen solution, corresponds to one rolling stock duty.

In the next section, we describe in more detail how paths are generated. Each iteration, the master problem uses the current set of generated paths and returns the least-cost solution to dissolve all unbalances, provided it can retrieve a feasible solution. Because paths are iteratively added, this solution approach is comparable to column generation. Therefore, the following mathematical formulation is referred to as the master problem.

Let i be an arbitrary path of the set I, which consists of all generated paths so far. The number of elements of set I only increases, because each iteration, at least one path is added to this set. A path i has multiple characteristics. First of all, a path i consists of an empty movement and/or a sequence of adjusted compositions compared to the initial schedule. The adjustments lead to a path which dissolves an unbalance between two rolling stock duties: $r_l \in B^1_{s,m}$ and $r_f \in B^2_{s,m}$. In other words, it forms a path from station $s_l = s_a(t^\infty_{r_l})$ to station $s_f = s_d(t^0_{r_f})$ for a given train unit of type $m \in M$.

The cost of a path is equal to the objective value of its corresponding shortest path, obtained from the time-dependent graph, and this is denoted as c_i. Furthermore, let $m_i \in M$ denote the selected rolling stock type of path i. Next, let s^+_i and s^-_i denote the station where path i starts and the station where the path ends respectively. With this information, one can identify that a path i was the shortest path of a time-dependent graph with corresponding triplet (s^+_i, s^-_i, m_i). A path could consist of possible adjustments to the initial schedule of trips. Let $b_{i,t} = 1$ if path $i \in I$ changes the composition of a trip $t \in T^1 \cup T^2$ and $b_{i,t} = 0$ otherwise. As stated in the previous section, no more than one train unit can be coupled to the initial composition of a trip. Also, only a maximum of one path, corresponding to a rolling stock duty r, can be selected to solve one unbalance. Let $a_{i,r} = 1$ if path $i \in I$ solves an unbalance by using a train unit of rolling stock duty $r \in R^1 \cup R^2$ and $a_{i,r} = 0$ otherwise. Notice that, for a given path i, it holds that $\sum_{r \in R^1, R^2} a_{i,r} = 2$. By way of explanation, a path i always corresponds to the shortest path from a rolling stock duty $r \in R^1$ to a duty $r \in R^2$. Finally, the decision variable x_i can be introduced. This binary decision variable is equal to 1 if path $i \in I$ is chosen in the solution and $x_i = 0$ otherwise. Then, the master problem becomes:

$$\min \quad \sum_{i \in I} c_i x_i \tag{14}$$

$$\text{s.t.} \quad \sum_{\substack{i \in I: \\ m_i = m}} x_i = \min \left\{ \sum_{s_1 \in S^+_m} b_{s_1,m}, - \sum_{s_2 \in S^-_m} b_{s_2,m} \right\}, \qquad \forall m \in M, \tag{15}$$

$$\sum_{\substack{i \in I: \\ s^+_i = s_1}} x_i \leq b_{s_1,m}, \qquad \forall s_1 \in S^+_m, m \in M, \tag{16}$$

$$\sum_{\substack{i \in I: \\ s^-_i = s_2}} x_i \leq -b_{s_2,m}, \qquad \forall s_2 \in S^-_m, m \in M, \tag{17}$$

$$\sum_{i \in I} a_{i,r}\, x_i \leq 1, \qquad \forall r \in R^1 \cup R^2, \tag{18}$$

$$\sum_{i \in I} b_{i,t}\, x_i \leq 1, \qquad \forall t \in T^1 \cup T^2, \tag{19}$$

$$x_i \in \mathbb{B}, \qquad \forall i \in I. \tag{20}$$

The objective function (14) minimises the costs of the selected paths. The constraints (15), (16) and (17) are similar to the mathematical formulation described in Section 6.1. The first restriction (15) imposes that every unbalance needs to be solved. Constraints (16) and (17) make sure that a path i respectively starts at a station with a surplus and ends at a station with a shortage. Restriction (18) specifies that a rolling stock duty cannot be selected more than once. Similarly, Constraint (19) imposes that two (or more) paths, which change the composition of a trip t, lead to a conflict. Therefore, only zero or one path, which changes the composition of this trip, can be chosen. Lastly, Constraint (20) imposes the decision variable x_i to be binary.

In the earlier phase, the number of generated paths could be low. If the formulation (14)-(20) does not lead to a feasible solution, then one needs to identify where the infeasibility is caused by. The infeasible constraint can be identified by including dummy variables to the formulation and/or analysing the output of the solver. Provided that at least one path $i \in I$ corresponds to a triplet (s_l, s_f, m), $\forall\ s_l \in S_m^+, s_f \in S_m^-, m \in M$, as explained in the following section, the infeasibility can only occur in the following two constraints. Here, we mention why the infeasibility occurs and what appropriate action can be taken to deal with the infeasible constraint:

- The infeasibility occurs in Constraint (18), then two (or more) paths cannot be selected, because this would lead to multiple usages of a rolling stock duty $r \in R^1 \cup R^2$. Assume that the two conflicting paths are $i_1, i_2 \in I$. The initial time-dependent graphs G_1, G_2 correspond to respectively the triplet $(s_{i_1}^+, s_{i_1}^-, m_{i_1})$ and $(s_{i_2}^+, s_{i_2}^-, m_{i_2})$. Then, both graphs G_1, G_2 need to remove the edge between the grey destination node and the node corresponding to rolling stock duty r. Afterwards, the shortest path of graphs G_1 and G_2 needs to be recomputed and the newly obtained shortest paths are included in the set I.
- The infeasibility occurs in Constraint (19), then, comparable to the previously infeasible constraint, two (or more) paths cannot be selected due to a duplicate usage of a certain trip t. Let i_1 and i_2 denote the paths which use trip t and let G_1 and G_2 denote the corresponding time-dependent graphs of these paths. Then, for both graphs, the edges representing trip t are excluded from the graph. Again, the distances of the vertices are recomputed and the new shortest paths are added to the set I.

Instead of only two conflicting paths i_1, i_2, there could be more than two incompatible paths. Then, more shortest paths need to be recomputed. On the other hand, if the mathematical formulation (14)-(20) leads to a feasible solution, then all unbalances are solved. Nonetheless, it is possible to generate more paths, even if the so-called master problem is feasible. By including more paths, the objective function can only decrease in value. The choice of continuing after a feasible solution is clarified in the next section.

6.3.2 Path Generation

With the described master problem, we can explain in more detail how the path generation approach can dissolve all unbalances. Recall that this approach is somewhat similar to column generation, because each iteration paths are generated until a stopping criterion has been met. The master problem corresponds to the linear program described in the previous section and the computation of the shortest path of a time-dependent graph can be considered as a subproblem. An important difference between our method and column generation is that the reduced costs of the dual problem are not used. Before we continue to the procedure of the path generation approach, let us first discuss how multiple paths can be generated from a single time-dependent graph.

Let us consider one specific time-dependent graph with corresponding triplet (s_l, s_f, m). With topologically sorting, the shortest path can be easily calculated. Let i_1 denote the shortest path and this path will be included in the set I. However, assume that $b_{s_l,m} = 2$ and $b_{s_f,m} = -2$. In this case, two units are in surplus at station s_l and station s_f has two units in shortage. This unbalance could be solved by generating two paths from s_l to s_f. The first shortest path i_1 was already computed. However, the graph needs to change in order to find a second, alternative path.

Section 6.2.4 already exposed how one can obtain multiple paths from a graph with multiple source and/or destination nodes, such that all unbalances between these two stations are dissolved. The first constructed path is indeed the shortest path i_1 of the graph. Before determining the second path, a slight adjustment to the graph is necessary. Let $v_1, v_2 \in V_l$ denote the source nodes of respectively rolling stock duties $r_1, r_2 \in R^1_m$. Assume the shortest path starts at source node v_1 and an arbitrarily edge $e_1 \in E$ is part of the shortest path i_1. In order to determine a new path, two actions can be taken: removing all outgoing edges of node v_1 and/or removing edge e_1 from the graph. Then, if the distances of all vertices are recomputed, a different 'shortest' path i_2 is obtained.

The choice, of either removing outgoing edges from a source node or removing an arbitrarily used edge, depends on the situation. If the second action is chosen, then it generally leads to a slightly different path. Moreover, chances are high that the path still starts at the same source node of the previously obtained path. It is convenient to perform this step, if one wants to mark a certain trip as (temporary) forbidden. Like explained in the previous section, this would be beneficial to use if multiple paths contain the same trip and, therefore, the master problem becomes infeasible. However, in other cases, it could be convenient to remove outgoing edges from one or multiple source nodes. To dissolve an unbalance, where $b_{s_l,m} = 2$ and $b_{s_f,m} = -2$, it is interesting to consider a path from each source node and destination node. Moreover, disregarding a source or destination node could also be applied when the infeasibility of the master problem is caused by conflicting rolling stock duties.

In Algorithm 2, a short overview of the path generation method can be found. Lines 1 - 3 initialise the algorithm. First, the initialisation starts by constructing the time-dependent graph of triplet (s_l, s_f, m), for every $s_l \in S^+_m,\ s_f \in S^-_m,\ m \in M$. Afterwards, the set of paths I is initialised. If more paths are included in this step, fewer iterations would be necessary, which eventually leads to a lower running time. Instinctively, the shortest path of each time-dependent graph (s_l, s_f, m) is included in this set. Aside from these paths, one could include multiple paths from a single time-dependent graph. Consider a triplet (s_l, s_f, m), where it holds that $b_{s_l,m} > 1$. Here, station s_l has multiple train units of type m in surplus. Then, by repeatedly removing the outgoing edges of the selected source node of the previous path, $b_{s_l,m}$ unique paths can be added to the set I. Equivalently, $|b_{s_f,m}|$ paths can be generated whenever $b_{s_f,m} < -1$.

Then, until a stopping criterion has been met, paths are iteratively added to the set of paths I. First, Line 5 states that the master problem is solved. Consequently, one of the following two situations occurs: the master problem is feasible or infeasible. If the master problem happens to be infeasible, one needs to identify the infeasible constraint as explained in Section 6.3.1. Here, either outgoing edges from a source or destination node, or specific edges corresponding to a trip are removed. Then, the shortest path of the temporary adjusted time-dependent graph is recomputed and the new path i is added to the set of paths I (Lines 6 - 9).

Algorithm 2: Solve Multiple Unbalances

```
1  Initialise all time-dependent graphs;
2  Initialise the set of paths I;
3  accept: ← FALSE;
4  while accept = FALSE do
5  |  Solve the masterProblem(I);
6  |  if infeasible then
7  |  |  Obtain a new pair (s_l, s_f, m);
8  |  |  i ← shortestPath(s_l, s_f, m);
9  |  |  I ← I ∪ i
10 |  else
11 |  |  accept ← stopCriterion();
12 |  |  if accept ← FALSE then
13 |  |  |  Find a new pair (s_l, s_f, m);
14 |  |  |  i ← shortestPath(s_l, s_f, m);
15 |  |  |  I ← I ∪ i
16 |  |  end
17 |  end
18 end
19 Add all deadheading trips to I;
20 Solve masterProblem(I);
```

On the other hand, if the master problem is feasible, we have found a feasible set of paths which dissolves all unbalances without violating any constraints. Then, the question rests whether one wants to improve the current solution or, alternatively, stops the algorithm. There are multiple options to be considered as a stopping criterion (Line 11). One could be satisfied with a feasible solution, but other options would be to stop when one cannot find an improvement of the objective function after n iterations or to disrupt the algorithm after t running time. Note that the objective value can only improve by generating more paths, because paths are never removed from the set I.

Lines 13-15 state that a new path must be found, that potentially improves the objective function. First, a time-dependent graph must be identified, whose shortest path does not exclusively include a deadheading trip. In other words, if the shortest path of the graph is a deadheading trip from s_l to s_f, this graph cannot produce paths that improve the objective function. There are two alternative options to determine the second-best shortest path. One could implement an algorithm for the k shortest path routing (Hershberger et al., 2007). A disadvantage of this approach is that topologically sorting cannot be used and one should switch to another algorithm, such as the Bellman-Ford algorithm (Bellman, 1958). Another, less efficient, method would be to stick to topological sorting. Here, one should temporarily eliminate each edge of the previous shortest path. The least-cost path among the resulting, alternative paths is the second-best shortest path.

Lastly, Lines 19 - 20 are reached when the stopping criterion has been met. Then, the master problem is called once again. Instead of only using the set of paths as I, also all deadheading trips between $s_l \in S_m^+$ and $s_f \in S_m^-$ are included (as a path). Including these deadheading trips could possibly increase the objective function and therefore, these are added to the set of paths. The last master problem returns the set of selected paths which dissolves all remaining unbalances.

6.4 Generate Alternative Inventory Levels

In this section, we describe how the end-of-day and start-of-day inventories can be changed and how changes to the initial (optimal) solution could lead to a better, balanced rolling stock schedule. In contrast to the previous section, the inventory levels are unfixed. The outlined neighbourhoods implement changes to the initial (near-)optimal schedules. These adjustments are not specifically intended to improve the objective function of a schedule of a single day, but to alter the end-of-day or start-of-day inventory.

Simulated annealing is used to diversify the solution space using the described neighbourhoods. By changing the inventory levels, the number of unbalances changes. In other words, at the costs of changing the (near-)optimal schedules, the intention is to achieve less unbalances. To be more precise, instead of attaining less unbalances overall, the goal is to decrease the (expected) cost to dissolve all remaining unbalances.

Eventually, the remaining unbalances of the altered inventory levels can be solved by using the solution approach described in Section 6.3 or the upper bound of Section 6.1. The combination of path generation and simulated annealing will be explained in Section 6.5. Nevertheless, Section 6.4.1 explains the general solution approach where simulated annealing uses various neighbourhoods. This iterative process randomly selects neighbourhoods until a stopping criterion has been met. Afterwards, Section 6.4.2 describes the defined neighbourhoods in more detail.

6.4.1 Simulated Annealing

Equivalently to previous solution approaches, the starting points are the (near-)optimal rolling stock schedules of two consecutive days x_1, x_2. Let $F(x_1)$ and $F(x_2)$ correspond to the costs of the schedule of respectively the first and the second day. Here, $F(x)$ is calculated as $F(X, Z, N)$ as denoted in Equation (5). At first sight, adjusting the initial rosters does not seem to be beneficial, because the schedules are assumed to be (near-)optimal. Any change to x_1 or x_2 leads to an increment of the objective function. Let C_{x_1,x_2} correspond to the estimated, additional costs of bringing the initial schedule in balance. Then, the total (estimated) costs of the merged rolling stock schedule becomes: $F(x) = F(x_1) + F(x_2) + C_{x_1,x_2}$.

If the additional cost-increase of adjusting the schedule x_1 (or x_2) is lower than the (estimated) cost-decrease of dissolving all unbalances, the adjustment will be accepted. In other words, the idea of this solution approach is to apply adjustments to the initial schedules such that it leads to a more preferred end-of-day and/or start-of-day inventory. An adjustment to a schedule will be executed if the expected reduction in costs C_{x_1,x_2} exceeds the costs of the adjustments in the initial schedule or otherwise be accepted with a certain probability. An adjustment to the initial schedule that does not change the end-of-day or start-of-day inventory, but increases the objective value, is in general, not a beneficial adjustment. However, temporary increments could prevent the solution from staying in local optima.

We use the neighbourhoods of simulated annealing to find a better solution. Here, each neighbourhood makes adjustments to the initial schedule with the goal to change the inventory levels. Simulated annealing allows for degradations in the objective value to incorporate diversification to the generated solutions. Furthermore, it uses a parameter, often referred to as the temperature T, to regulate the probability of choosing a worse solution. If the temperature T is larger, then the probability of choosing a worse objective value gets larger. Generally, the temperature slowly declines during simulated annealing.

Each neighbourhood changes either the schedule of the first x_1 or the second day x_2. Let $N(x_1)$ denote a neighbourhood that changes the schedule of the first day and let $N(x_2)$ be a neighbourhood that adjusts the schedule of the succeeding day. If we consider $N(x_1)$, only an adjustment to the end-of-day inventory could lead to a (potential) decline of C_{x_1,x_2}. Similarly, only an adjustment to the start-of-day inventory of the schedule of the second day x_2 could change C_{x_1,x_2}. In conclusion, cost-increasing changes to the initial (near-optimal) schedules are necessary, provided the altered end-of-day and start-of-day inventories lead to an expected cost-reduction of C_{x_1,x_2}.

In Algorithm 3, an overview of simulated annealing can be found. Lines 1 and 2 initialise the heuristic. Here, x represents the initial, unbalanced schedule of two consecutive days. Furthermore, the starting temperature T_0 is set to its initial value. Then, Line 4 and 5 state that the day to be considered and the neighbourhood $N(x)$ are randomly selected. In the following section, the usage of the neighbourhoods is explained in more detail. A random candidate solution is selected from this neighbourhood, provided it does not validate any restrictions (Line 6). This candidate solution x^* will be implemented to the initial schedule if the (expected) costs $F(x^*)$ are lower than the costs of the initial schedule $F(x)$ (Line 7-8). Otherwise, Lines 10-13 state that a random number p is selected and that the candidate solution will be selected if $p < \exp\left(\frac{F(x)-F(x^*)}{T}\right)$.

Algorithm 3: Simulated Annealing

```
   Result: A schedule x* with alternative start-of-day and end-of-day inventories
 1 x ← Initial schedule;
 2 Set T ← T_0;
 3 while accept = FALSE do
 4 |   Randomly pick schedule x_1 or x_2;
 5 |   Pick a random neighbour N(x);
 6 |   Select (randomly) a candidate x* ∈ N(x);
 7 |   if F(x*) < F(x) then
 8 |   |   x ← x*;
 9 |   else
10 |   |   Select random number p ∈ [0, 1);
11 |   |   if p < exp((F(x)−F(x*))/T) then
12 |   |   |   x ← x*;
13 |   |   end
14 |   end
15 |   Update temperature T;
16 |   accept ← stoppingCriterion();
17 end
```

At Line 15, the temperature will be updated. In general, the temperate declines every iteration. If the temperature is high, the probability of selecting a schedule which worsens the objective value is higher. It could be beneficial to initially start with diversifying the given schedule before gradually decreasing the objective function. The choice of the starting temperature T_0, the updates and the stopping criterion (Line 16) could differ for every problem (and instance). Here, the temperature is updated as follows: $T \leftarrow \alpha\, T$, where $0 < \alpha < 1$. Moreover, the current schedule is accepted if the algorithm does not lead to any changes in the schedule x for at least β iterations.

Lastly, note that C_{x_1,x_2} is an estimation of dissolving the unbalance between two schedules x_1 and x_2. These costs can be estimated using the path generation approach as described in Section 6.3. However, the main disadvantage of using this method, is that it is relatively time-consuming. C_{x_1,x_2} needs to be calculated every iteration and, therefore, a faster estimation would be more efficient. Moreover, a schedule with total costs $F(x^*)$ is selected based on a probability. Hence the quality of the costs can be an estimation and, therefore, C_{x_1,x_2} can be equal to the upper bound of all unbalances, as described in Section 6.1.

6.4.2 Neighbourhoods

This section introduces the so-called neighbourhoods. These neighbourhoods execute at least one adjustment to the initial schedule. A neighbourhood explores their predefined solution space and tries to find a solution with a lower objective function. Changes to the initial schedule x could lead to a difference in the total costs $F(x)$. The difference in costs can be computed by comparing the costs of every altered trip

using Equation (6). Then, adding each difference in costs leads to the total difference in costs between the adjusted and the initial schedule $F(x_1^*) - F(x_1)$ or $F(x_2^*) - F(x_2)$. If the neighbourhood alters the start-of-day or end-of-day inventory, the difference in (expected) costs to solve the remaining unbalances can be calculated as $C_{x_1^*,x_2} - C_{x_1,x_2}$ or $C_{x_1,x_2^*} - C_{x_1,x_2}$. Here, the estimated costs of dissolving all unbalances C_{x_1,x_2} can be computed using the mathematical formulation of Section 6.1 or one could even use a greedy approach to estimate C_{x_1,x_2}. The total difference in costs between the altered and the initial schedule is presented as $F(x^*) - F(x)$.

Some of the below-described neighbourhoods do not particularly search for a better solution, but are used to diversify the current schedule. For every neighbourhood, we describe in a precise way which adjustments are applied to the initial schedule. If the neighbourhood alters the end-of-day or start-of-day inventory, then this change is explicitly mentioned. Furthermore, notice that each neighbourhood can only adjust the schedule of either x_1 or x_2. Depending on the selected schedule, the neighbourhood could have a slightly deviating approach. This is because our intention is to alter the end-of-day inventory of schedule x_1 and, on the contrary, to change the start-of-day inventory of schedule x_2. Lastly, if a neighbourhood can only be executed under some conditions, then these requirements are explicitly noted.

Decoupling the train unit earlier The idea of this neighbourhood is to decouple earlier than planned, such that the train unit arrives at a different station. Let $t_{k(1)}, t_{k(2)}, ..., t_{k(|K_r|)}$ be the sequence of trips of a rolling stock duty $r \in R^1$. Then, after performing a certain trip $1 \leq n < |K_r|$, the train unit of type m_r is decoupled from its initial composition. Notice that the composition of the remaining trips $t_{k(n+1)}, ..., t_{k(|K_r|)}$ must contain at least one train unit, such that every trip is covered by a non-empty composition. By applying this neighbourhood, the inventory of train unit type m_r changes. The initial destination station of rolling stock duty $r \in R^1$ is $s_a(t_r^\infty)$. However, this destination station changes to a new end station $s_a(t_r^\infty)^*$, which is the arrival station of trip $t_{k(n)}$. In conclusion, the adjustments to the end-of-day inventory can be summarised as:

- $I^\infty_{s_a(t_r^\infty),m_r} \leftarrow I^\infty_{s_a(t_r^\infty),m_r} - 1$;
- $I^\infty_{s_a(t_r^\infty)^*,m_r} \leftarrow I^\infty_{s_a(t_r^\infty)^*,m_r} + 1$.

Coupling the train unit later A similar idea can be used to change the start-of-day inventory of the succeeding day. Instead of decoupling earlier than planned, the train unit can be coupled later to rolling stock duty $r \in R^2$ than planned. In this case, the train unit 'misses' the first trips. Here, the initial departure station of rolling stock duty $r \in R^2$ is $s_d(t_r^0)$. If the train unit is coupled before trip $1 < n \leq |K_r|$ departs, then the new departure station becomes $s_d(t_r^0)^* = s_d(t_{k(n)})$. Similarly, the start-of-day inventory changes to:

- $I^0_{s_d(t_r^0),m_r} \leftarrow I^0_{s_d(t_r^0),m_r} - 1$;
- $I^0_{s_d(t_r^0)^*,m_r} \leftarrow I^0_{s_d(t_r^0)^*,m_r} + 1$.

Cancel initial (de)coupling movements In this neighbourhood, the idea is to simply remove shunting movements of the initial rolling stock schedule. Here, the focus lies on possibly changing the end-of-day or start-of-day inventory, but, more importantly, an attempt to diversify the schedule. In the initial schedule, decoupling movements are supposed to decrease the number of 'unnecessary' carriage kilometres. If one removes an initial shunting movement, then the above-described neighbourhoods could search for an alternative shunting movement. Therefore, by first removing and then subsequently adding a decoupling

movement, the number of unnecessary carriage kilometres is limited while the inventory level possibly changes. When considering this neighbourhood, one should especially pay attention whether the altered compositions remain feasible.

Change the train unit type of a rolling stock duty This neighbourhood changes the rolling stock type of an entire rolling stock duty. Initially, the type of the rolling stock duty $r \in R$ is $m_r \in M$. The idea is to change the type of this duty to a train unit type $m_r^* \in M$, where $m_r \neq m_r^*$. In other words, the composition of every trip of the duty is adjusted to a new composition, where one train unit of type m_r is replaced by m_r^*. By using this neighbourhood, the start-of-day as well as the end-of-day inventory of a day changes. Therefore, this method can be used to schedule x_1 as well as to schedule x_2. This method leads to the following changes of the inventories:

- $I^0_{s_d(t^0_r),m_r} \leftarrow I^0_{s_d(t^0_r),m_r} - 1$;
- $I^\infty_{s_a(t^\infty_r),m_r} \leftarrow I^\infty_{s_a(t^\infty_r),m_r} - 1$;
- $I^0_{s_d(t^0_r),m^*_r} \leftarrow I^0_{s_d(t^0_r),m^*_r} + 1$;
- $I^\infty_{s_a(t^\infty_r),m^*_r} \leftarrow I^\infty_{s_a(t^\infty_r),m^*_r} + 1$.

This neighbourhood is especially effective if, considering the first day, station $s_a(t^\infty_r)$ has a surplus in type m_r and a shortage in type m_r^*. In this case, the number of unbalances is reduced by two.

However, it is relatively difficult to find an available train unit m_r^*. The majority of the train units already belongs to a rolling stock duty or is in maintenance. Especially on weekdays, the number of unused train units is low. If a train unit of type m_r^* is not available, one could interchange two rolling stock duties. Let r_1 and r_2 be two rolling stock duties with respectively type m_r and m_r^*. Provided it does not violate any constraints, duty r_1 changes to type m_r^* and rolling stock duty r_2 changes from type m_r^* to m_r. Performing such an interchange leads to several changes in the start-of-day and end-of-day inventory. If chosen perfectly, one interchange between two duties could even dissolve four unbalances at once.

A disadvantage of the described neighbourhood is that the start-of-day inventory of x_1 or the end-of-day inventory of x_2 changes. This does not lead to any problems when only two days are considered, but it lead to difficulties when more than two consecutive days are studied. Suppose that the schedule between the second and the third day is already in balance, while there are still unbalances between the first and the second day. Then, a change to the end-of-day inventory of the second day causes an unbalance between the already balanced second and third day. Therefore, one could impose that it is only possible to (inter)change the rolling stock type of an entire duty, provided it does not lead to any changes in the start-of-day inventory of x_1 or the end-of-day inventory of schedule x_2.

Change the train unit type of a fraction of a rolling stock duty This neighbourhood is similar to the one described above. However, instead of replacing all trips by a different train unit type, only a small fraction of the rolling stock duty is replaced. Let $\{t_{k(1)}, t_{k(2)}, ..., t_{k(|K_r|)}\}$ be the set of trips of a rolling stock duty $r \in R^1$ with train unit type m_r. Then, starting from trip $t_{k(n)}$, the initial train unit type m_r is replaced by type m_r^*.

Simultaneously decoupling a train unit m_r and coupling a unit m_r^* is assumed to be impossible. However, if trip $t_{k(n)}$ is not a predefined successor of trip $t_{k(n-1)}$, the change is possible, provided the new rolling stock type m_r^* is available at station $s_d(t_{k(n)})$. In other words, one could perform this movement if the train unit m_r^* of duty r is in storage at station $s_d(t_{k(n)})$. Between the two peak hours, multiple train units are temporarily unused and moved to the storage of a station. If both rolling stock types m_r and

m_r^* are available before the next trip $t_{k(n)}$ departs, the initial train unit m_r can be replaced by m_r^*. Then, this leads to the following changes:

- $I^{\infty}_{s_a(t^{\infty}_r),m_r} \leftarrow I^{\infty}_{s_a(t^{\infty}_r),m_r} - 1$;
- $I^{\infty}_{s_a(t^{\infty}_r),m^*_r} \leftarrow I^{\infty}_{s_a(t^{\infty}_r),m^*_r} + 1$.

The advantage of this neighbourhood is that it does not change the start-of-day inventory of the first day nor the end-of-day inventory of the second day. Similarly, the same change can be applied to the succeeding day. In this case, the idea is to change the start-of-day inventory instead of the end-of-day inventory. Then, starting from trip $t_{k(1)}$ till trip $t_{k(n-1)}$, the initial type m_r is replaced by rolling stock type m_r^*. Thus, this leads to the following adjustments:

- $I^{0}_{s_d(t^{0}_r),m_r} \leftarrow I^{0}_{s_d(t^{0}_r),m_r} - 1$;
- $I^{0}_{s_d(t^{0}_r),m^*_r} \leftarrow I^{0}_{s_d(t^{0}_r),m^*_r} + 1$.

In both cases, note that the train unit m_r^* is available at the station and it remains unused for the rest of the day. If, in the initial schedule after mid-day, trips are assigned to the swapped train unit m_r^*, the remaining trips can be executed by the train unit m_r. Similarly to the previous described neighbourhood, the rolling stock duties can interchange their assigned rolling stock. Instead of interchanging the train units for all trips of each rolling stock duty, only a fraction of the trips is swapped.

Change the mid-day inventory The previous neighbourhood also mentioned that some train units are unused between the two peak hours. These units are stored at a station and coupled to a trip just before the peak hour starts. The 'noon' inventory is always in balance, because, otherwise, the initial schedule would be infeasible. However, similar to the dissolving unbalances in the night, one could make adjustments to the mid-day inventory. The noon inventory could be changed by using the previously mentioned neighbourhoods, such as decoupling or changing the type. The goal of this neighbourhood is to search for an alternative mid-day inventory which could eventually change the end-of-day inventory of x_1 or the start-of-day inventory of x_2.

To clarify this neighbourhood, suppose we have the following three rolling stock duties:

1. A train unit of type m_r starts at station A. It is in storage at station A during noon and ends at station A.
2. A train unit of type m_r starts at station B. It is in storage at station B during noon and ends at station B.
3. A train unit of type m_r^* starts at station B. It is in storage at station B during noon and ends at station C.

Suppose that all unbalances are solved if type m_r^* changes its destination station from station C to station A and if one train unit type m_r changes its destination from station A to station C. At first sight, the rolling stock types of duties 1 and 3 could be swapped and the problem would be solved. However, assume that this is extremely costly and/or the start-of-day inventory should be fixed. Then, this neighbourhood is particularly useful. Suppose that duty 1 could also be stored at station B during noon. In that case, only a fraction of the trips of the rolling stock duties 1 and 3 could be swapped and the unbalance would be dissolved. Similarly, suppose that duty 2 could change its destination to station C. In that case, duties 2 and 3 could be partly swapped with each other. In conclusion, changing the mid-day inventory by applying other neighbourhoods could eventually change the end-of-day inventory or, on the contrary, the start-of-day inventory.

Change position within a composition Each task $k(i), i \in \{1, 2, ..., |K_r|\}$ represents a pair $(t_{k(i)}, q_{k(i)})$ which corresponds to respectively the trip and its position in the composition of that trip. In this neighbourhood, the goal is to diversify the solution by changing the initial position of a train unit. Technically, the transformation of the order of a composition does not lead to additional costs. The number of carriage kilometres, shunting movements and seat shortages remain the same. Furthermore, changing the order of a composition does not change the start-of-day or end-of-day inventory. However, by adjusting the composition, new shunting possibilities could arise. At certain stations, it is only possible to (de)couple at the rear or at the front of a train. In the initial schedule, it could be impossible to decouple a certain unit at a station, but by changing the order of a composition, this shunting movement could suddenly become executable.

6.5 Overview

In this section, we explain how the previously described methods can be integrated into one overall solution approach. The method of Section 6.3 can only dissolve unbalances between predetermined end-of-day and start-of-day inventories. This method cannot directly change inventory levels and, therefore, its solution space is relatively small. However, by definition, it does provide a better objective value than the upper bound of Section 6.1. The solution approach based on simulated annealing of Section 6.4 generates different end-of-day and start-of-day inventories. Its solution space is larger due to the introduced neighbourhoods. Both solution approaches are a self-contained method to solve the rolling stock rebalancing problem.

However, the prior outlined techniques can be used in an iterative process. By using an iterative procedure, it uses the random nature of simulated annealing and the relatively low computation time of path generation perfectly. Simulated annealing randomly generates different start-of-day and end-of-day inventories and the remaining unbalances can be solved using the solution approach described in Section 6.3. The overall solution approach can be outlined as follows:

Step 1: Set $F_{best}(x) \leftarrow \infty$.

Step 2: Obtain the (near-)optimal, independent schedules of the first day x_1 and the succeeding day x_2.

Step 3: Apply simulated annealing (Algorithm 3) with the initial schedules x_1, x_2 as input. Let x_1^* and x_2^* be the new schedules.

Step 4: Solve the remaining unbalances between x_1^* and x_2^*, using Algorithm 2. Let $F(x^*)$ be the obtained solution value of the new (balanced) schedule.

Step 5: If $F_{best}(x) > F(x^*)$, then set $F_{best}(x) = F(x^*)$. If a stopping criterion has been met, STOP. Else, return to Step 3.

Likewise other methods, the input of the integrated solution approach is the rolling stock schedule of the first x_1 and the second day x_2. The independently computed schedules are calculated using the composition model of Fioole et al. (2006). Although one could also use another method to compute these schedules, the assumption is that the assembled schedules are (near-)optimal. The reason is that this overall solution approach only tries to solve the unbalance in a least-cost manner and does not explicitly try to improve the solution value of the initial schedules. Step 1 and 2 of the described procedure serve as the initialisation of the overall solution approach.

Next, Step 3 initiates simulated annealing as explained in Section 6.4. It uses the independently calculated schedules as input and the goal is to reduce the total costs of these schedules. Although the schedules are (near-)optimal, the costs of dissolving all unbalances are, in practice, relatively high. The idea of the iterative process is that simulated annealing randomly produces different start-of-day and end-of-day inventories. The remaining unbalances between the altered schedules are solved in Step 4 by using the path generation approach. Because simulated annealing is a probabilistic technique, the altered end-of-day and start-of-day inventory is, in general, different for each iteration. Hence, due to the random nature, the remaining unbalances are different for each iteration and, consequently, the path generation approach also returns a slightly different solution.

Finally, Step 5 updates the (provisional) solution value found so far. The iterative process continues until the stopping criterion has been met. A logical criterion would be to stop the algorithm if it reaches a certain time limit. Nonetheless, one could also terminate the algorithm if the objective value is sufficient or after a certain number of iterations.

7 Results

This chapter reveals the results of the previously described solution approaches. Both the solution quality and computation time are the two major points of interest. First, Section 7.1 introduces the used instances for the computational experiments. Then, Section 7.2 illustrates the results of the sequential solution approach and Section 7.3 reports the outcomes of the three described heuristic approaches. Lastly, in Section 7.4, we perform a sensitivity analysis.

7.1 Data

NS provided all the input required for the rolling stock rebalancing problem: the railway timetable, rolling stock characteristics, shunting possibilities, (dis)allowed compositions, etc. Here, we examine various cases with different sets of rolling stock and we consider different days. Logically, during the weekend, the usage of the rail network is less dense than on weekdays. On the other hand, due to the lower expected number of passengers during the weekend, fewer trains are used. Before introducing the instances, let us highlight a major difference in rolling stock.

The rolling stock of NS can be classified into two principal categories: *Sprinters* and *Intercities*. The first category is used for shorter distances and, generally, stops at every station the train passes. It is used to connect the local stations to the larger stations. *Sprinters* often traverses back and forth between two stations. Moreover, this train unit does not execute many shunting movements in comparison to the *Intercities*. Therefore, a *Sprinter* can also be seen as less flexible than the *Intercity*. The *Intercity* is particularly suitable to travel larger distances. During rush hours, multiple *Intercities* are coupled to each other to cover the expected rise of passengers. After rush hours, these extra train units are decoupled and stored at a station. Moreover, an *Intercity* generally skips the smaller stations.

Table 1: List of all examined instances with their primary characteristics.

Instance	# Trips	# Types	# Available Train Units	# Initial Unbalances	# Stations	Rolling Stock Type	Days
1	1482	2	52	3	17	*Flirt*	Sat. & Sun.
2	1643	2	52	15	17	*Flirt*	Mon. & Tue.
3	6780	9	308	27	83	*Sprinters*	Sat. & Sun.
4	9138	9	308	72	83	*Sprinters*	Tue. & Wed.
5	9149	9	308	76	83	*Sprinters*	Mon. & Tue.
6	8342	8	351	65	60	*Intercities*	Sat. & Sun.
7	9768	8	351	110	60	*Intercities*	Tue. & Wed.
8	9823	8	351	99	60	*Intercities*	Mon. & Tue.
9	15776	21	695	125	104	*All*	Sat. & Sun.
10	19628	21	695	186	104	*All*	Mon. & Tue

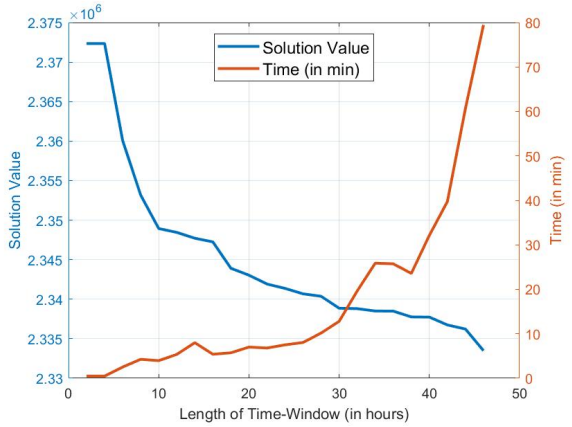
To interpret the different instances, take a look at Table 1. This table reveals the instances and their principal characteristics. It describes respectively the number of trips, the number of unique rolling stock types, the total number of available train units, the initial number of unbalances and the total number of visited stations. Note that the number of initial unbalances is strictly smaller than the number of available train units. The last two columns describe respectively the used rolling stock type and the examined days. The rolling stock type *Flirt* represents a subset of two types out of all available *Sprinters*. *All* represents all available rolling stock types: *Sprinters*, *Intercities* and other, less common rolling stock types (i.e. *ICE* or *TGV*).

Before moving on to the results, let us discuss the predefined values of the parameters. Equation (5) introduced the weighted parameters for the principal objectives: C_{CKM}, C_{SSK} and C_{SHM}. These objectives denote respectively the carriage kilometres, the seat-shortage kilometres and the number of shunting movements. These parameters are set to $C_{CKM} = 0.1$, $C_{SSK} = 0.5$ and $C_{SHM} = 10$. Furthermore, the penalty factor for deadheading movements is set to $\gamma = 5$, unless stated otherwise.

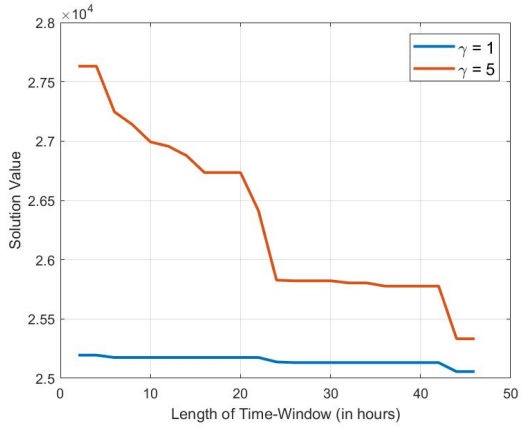
7.2 Sequential Solution Approach

The results of the sequential solution approach of Chapter 5 are illustrated in Figure 6. Here, the focus lies on the quality of the solution and, perhaps more importantly, the running time. We start by examining Figure 6a, which is based on instance 8. This instance represents the rolling stock schedule of Monday and Tuesday for all *Intercities*. The graph illustrates the solution value and computation time for different specified time windows. The smallest time window is only 2 hours, where every departure, before Tuesday 02:00 and after Tuesday 04:00, is fixed. Then, the time window is slightly increased by two hours, such that $[\tau_1, \tau_2] = [01{:}00, 05{:}00]$. The time window is repeatedly expanded by 2 hours in a similar fashion until all trips are unfixed. In the latter case, the time window lasts 46 hours. Here, all trips between Monday 04:00 and Wednesday 02:00 are unfixed.

Note that Figure 6a has two different y-axes. The left y-axis illustrates the solution value for different lengths of the time window and the right y-axis shows the corresponding computation time in minutes. Clearly, the solution value is non-increasing when the length of the time window increases. More trips are unfixed when the length of the time window expands and, therefore, the solution value can only decrease. However, considering more trips generally tends to lead to a higher computation time. Although the running time sporadically decreases when the time window expands, there is a clear increasing trend visible. The running time does even seem to grow exponentially. Accordingly, Figure 6a clearly expresses the trade-off between running time and the quality of a solution.



(a) This figure plots the length of the specified time window against the solution value and the running time.



(b) Comparison of two different values for the deadheading penalty factor γ against the length of the time window. The running time is negligible.

Figure 6: Results of the sequential solution approach for respectively instance 8 and 2.

Then, take a look at Figure 6b representing instance 2, which is the rolling stock schedule of type *Flirt* for Monday and Tuesday. The graph illustrates the solution value for different specified time windows for two predetermined penalty factors of deadheading trips $\gamma \in \{1, 5\}$. Equivalently to Figure 6a, a time

window of 2 hours fixes the composition of all trips departing before 02:00 and after Tuesday 04:00. Then, an expansion of the time window of two hours means that both boundaries are extended by one hour.

As can be seen in Figure 6b, the solution value decreases when the time window increases. The instance is relatively small and the running time is just a few seconds, even for the largest time window. However, let us focus on the solution value for various time windows. Notice that the objective value slowly decreases and, especially when $\gamma = 1$, barely decreases at all. It is also remarkable that the solution value does not seem to (substantially) improve for time windows larger than 24 hours. Even more remarkable is the fact that at a time window of 42 hours with $\gamma = 1$, the solution value (25,132) is (in percentages) closer to the upper bound (25,195) than to the optimal solution value (25,056). This sudden drop of the objective value is also visible in Figure 6a. Here, it would be beneficial to change to the train unit type of an entire rolling stock duty. Because at least one trip of the rolling stock duty remains fixed, this adjustment of rolling stock type is not possible.

7.3 Heuristic Approach

Throughout this section, we will present the results of the described methods of Chapter 6. Here, the focus lies mostly on the overall solution approach of Section 6.5. This method will be compared to the lower and upper bounds, the optimal solution and the self-contained solution approaches of Section 6.3 and Section 6.4. For the remainder of this section, the lower bound is abbreviated as LB and this is the summation of the objective values of the independently, computed schedules: $LB = F(x_1) + F(x_2)$. UB denotes the upper bound as described in Section 6.1. Here, the upper bound is computed based on the start-of-day and end-of-day inventories of the initial schedules x_1 and x_2. The optimal schedule is denoted as *Opt.* Throughout this section, the optimal objective value is presented as $-$, if the composition model cannot find the objective value within 30 minutes, provided the solution gap is not higher than 0.01%. *Unb.* represents the solution approach of Section 6.3 and is based on the unbalances of the initial rolling stock schedule. Simulated annealing is abbreviated as *SA*. First, simulated annealing is applied to the initial schedule and afterwards, the remaining unbalances are dissolved by only using deadheading movements. Finally, *Heur.* will denote the solutions of the merged solution approach, as explained in Section 6.5.

Before the results can be presented, let us discuss the parameter settings for the heuristic solution approaches. The solution approach of simulated annealing requires several predetermined parameters. The initial starting temperature T_0 is set to 100. The temperature is updated in the following way: $T \leftarrow \alpha T$, where $\alpha = 0.95$. The algorithm stops if none of the neighbourhoods can find any improvement on the objective. On the other hand, the path generation approach stops whenever the master problem finds a feasible solution. Lastly, let the time it takes to obtain the (independently calculated) rolling stock schedules of the instances be denoted as t^*. Then, the time limit of the overall solution approach is set to t^*.

Using the introduced instances and the predetermined parameters, we can finally compare the methods. The results can be seen in Table 2. For each method, the objective value and the running time in seconds are denoted. The lower bound of each instance is set to 100 and each objective value of the table is normalised to this lower bound. The running time of the lower and upper bound is not of interest and therefore omitted.

Table 2: Comparison of the described solution methods for different instances. Here, the solution value and, if noteworthy, the computation time (in seconds) are presented.

Inst.	**LB**	**Opt.**		**Heur.**		**SA**		**Unb.**		**UB**
		Obj	*Time*	*Obj*	*Time*	*Obj*	*Time*	*Obj*	*Time*	
1	100	104.3	2.4	104.3	15.4	104.6	0.3	104.3	0.2	104.6
2	100	101.5	2.6	102.4	18.6	104.1	0.6	109.7	0.2	110.6
3	100	100.9	932.8	102.9	110.6	103.2	9.5	105.1	0.6	105.3
4	100	100.3	1418.4	100.7	164.5	101.2	16.7	101.3	0.4	101.5
5	100	100.2	1711.4	100.7	197.0	100.9	18.9	100.9	0.7	101.0
6	100	101.1	1673.6	107.5	229.0	109.9	16.6	108.6	4.9	111.2
7	100	-	-	101.9	265.1	103.2	20.7	102.4	11.4	103.7
8	100	-	-	100.9	273.5	101.5	20.8	101.2	10.4	101.9
9	100	-	-	104.7	761.9	106.3	61.5	109.0	19.6	110.1
10	100	-	-	101.0	1163.9	101.3	166.7	101.2	27.6	101.4

The first two instances are relatively small and their optimal solution is found within seconds. But, as can be seen from Table 2, the optimal solution cannot be found for every instance within 30 minutes. Take a look at the upper bounds. The percentage gap to the lower bound differs for each instance, but for instances representing the weekend (instance 1, 3, 6, 9), this gap seems to be higher compared to the other instances. This is because the number of actually used train units is lower and, therefore, the costs of deadheading trip is relatively high compared to the considerably lower objective value. Moreover, seat-shortage kilometres are heavily penalised in the objective function. If one assigns an extraordinarily low number of units to a trip with a high number of expected passengers, the value of the objective increases substantially to twice or even several times the initial solution value. In the weekend, there is not really a rush hour and seat shortages happen sporadically, even if only one train unit is assigned to every trip.

The path generation approach is relatively fast, considering it only takes less than 30 seconds to solve all unbalances of the largest instance. By definition, the solution value of this method is lower than the upper bound. The difference between *Unb.* and *UB* falls between 0.1% and 2.6%. Especially for the larger instances, the path generation approach performs decently. For the instances of the *Intercities*, the number of shunting possibilities is higher and the method can easily find a better, least-cost alternative for the expensive deadheading movements. Lastly, notice that this method accomplishes to retrieve the optimal solution of instance 1.

Simulated annealing is slightly slower in terms of computation time compared to the path generation approach. Especially when the instances gets larger, the difference in running time between the methods increases. Nevertheless, in most cases, an alternative inventory level is obtained within a minute. The results, compared to the path generation approach, are remarkable. In 50% of the cases, simulated annealing retrieves a better solution value than the path generation approach. Therefore, simulated annealing does not seem to dominate the path generation approach in terms of solution quality or vice versa. Simulated annealing performs particularly better for the smaller instances or instances corresponding to the weekend compared to path generation.

Lastly, the overall solution approach retrieves, as expected, the best solution values. The running time of this method is set to t^*, the time it takes to determine the initial (independently calculated) rolling stock schedules. Therefore, the computation time of this method will not increase exponentially compared to the running time of a schedule of a single day. The quality of the overall solution approach is decent, but less effective for instances considering the weekend. For instance 1, 3, 6 and 9, the relative gap between this approach and the lower bound is still a few percent, whereas the gap of the other instances is close to, or even lower, than 1%.

Next, we assume that the start-of-day inventory of the first day and the end-of-day inventory of the succeeding day are fixed. In practice, these inventories cannot change, because the schedule before and/or after that day is already determined and in balance. Here, the obtained schedule of the following results must adhere to two conditions: the start-of-day inventory must be equal to the initial one of x_1, and the end-of-day inventory equals the end-of-day inventory of the second day x_2. These inventory levels are fixed and every solution method does not deviate from the fixed inventories. The results are shown in Table 3. In general, the running times of every method is equal, or similar, to the computation times of Table 2. Therefore, the computation times are not explicitly mentioned.

Table 3: Comparison of the described methods where the start-of-day and end-of-day inventory are fixed.

Inst.	**LB**	**Opt.**	**Heur.**	**SA**	**Unb.**	**UB**
1	100	104.3	104.3	104.6	104.3	104.6
2	100	103.5	103.7	105.3	109.7	110.6
3	100	102.4	103.0	103.4	105.1	105.2
4	100	100.5	100.8	101.3	101.3	101.5
5	100	100.6	100.8	100.9	100.9	101.0
6	100	102.5	108.1	108.3	108.6	111.2
7	100	-	101.9	102.4	102.4	103.7
8	100	-	100.9	101.6	101.2	101.9
9	100	-	107.0	108.5	109.0	110.1
10	100	-	101.0	101.3	101.2	101.4

First, notice that the solution values of the path generation and the upper bound are equal to the values of Table 2. These methods cannot change the start-of-day or end-of-day inventory of respectively x_1 or x_2. Furthermore, note that the gap between the optimal solution and the lower bound has increased. In Table 2, the optimal solution approach also adjusts these inventory levels. Consequently, the new constraint is included in the composition model and contributes to a slightly higher solution value.

Next, take a look at the results of *SA*. Although we expect the results of this column to be higher than the results of Table 2, this is not always the case. The values of instances 7 and 8 are lower when the start-of-day and end-of-day inventory are fixed. This is due to the reason that simulated annealing is a probabilistic technique. Due to its random nature, the solution value of this method differs each execution. It is noteworthy to mention that the neighbourhood, which (inter)swaps entire rolling stock duties, is considerably less used in comparison to the results of Table 2.

The randomness of simulated annealing seems to vanish when we compare the results of the overall heuristic approach. For each instance, the objective values are higher compared to Table 2. Nevertheless, the difference is in most cases less than one percentage point. The objective values of instance 1, 7, 8 and 10 are even equal to each other. For larger instances, the number of available units is higher and therefore, the number of units starting or ending at a certain station is higher. As a result, the possibility

of (inter)changing rolling stock duties is higher. Therefore, the restriction of a fixed start-of-day or end-of-day inventory affects the results, although this influence is relatively low.

Let us examine some remarkable notes about the previous results. First of all, take a look at Table 4. Here, the number of iterations of the overall heuristic approach and the number of iterations of simulated annealing are presented. As the instances become larger, the number of iterations for simulated annealing increases. It highly depends on the instance which neighbourhood is implemented more often and which one is barely used. For the smaller instances, the most used neighbourhoods are: decoupling the train unit earlier, coupling the train unit later and cancelling initial (de)coupling movements. The number of unique train unit types is low and an (inter)change between rolling stock duties is barely used. As the number of used *Sprinters* (instance 3-5) increases, the last-mentioned neighbourhood is used more often.

Table 4: The number of iterations of respectively the overall heuristic approach and simulated annealing.

Inst.	**# Iterations Heur.**	**# Iterations SA**
1	37	20
2	43	33
3	16	53
4	13	47
5	11	45
6	9	41
7	18	52
8	10	61
9	7	87
10	7	125

Nevertheless, *Sprinters* are usually not decoupled during noon and, therefore, neighbourhoods which change the mid-day inventory are sporadically used. For the larger instances (6-10), which include *Intercities*, nearly all neighbourhoods are at least once implemented. Especially neighbourhoods considering the mid-day inventory are implemented more often. The neighbourhood, which changes the position within a composition, is infrequently used and its purpose of diversifying the solution is unfortunately not noticeable.

In Table 4, we can also see the number of iterations for the overall solution approach. This number slowly declines as the instances get larger. The running time of the path generation and, principally, simulated annealing increases, if the number of trips of an instance increases. The increased running time of simulated annealing is caused by two reasons. First of all, the number of possibilities to change the schedule increases as the number of trips of an instance increases. Secondly, more possibilities lead to an increase in the objective and this leads to more iterations. So, not only does the number of iterations increase, but also the time per iteration. Because the time limit of the overall solution approach is set to t^*, the number of iterations decreases as the size of the instances increases.

To evaluate the randomness of simulated annealing, take a look at Figure 7. In this histogram, approximately 200 iterations of the overall heuristic approach are executed. On the x-axis, the solution values, normalised to the lower bound, are displayed and the frequency of the bins can be seen on the y-axis. First, simulated annealing is applied and, afterwards, the remaining unbalances are resolved by path generation. In other words, the solution values are based on Step 3 and 4 of the overall heuristic approach of Section 6.5. Due to the random procedure of simulated annealing, the succeeding path generation approach retrieves a different solution to dissolve the remaining unbalances.

As can be seen from Figure 7, this distribution of different solution values is caused by the randomness of simulated annealing. However, the difference between the highest and lowest solution value is only 0.7 percentage point. Furthermore, more than 95% of the solution values fall within a gap of 0.5 percentage point. The repetitive procedure of the overall solution approach has its advantages due to the random nature of simulated annealing. Nonetheless, executing for example 20 or 200 iterations does not seem to lead to a major discrepancy for the objective value, because the number of outliers is relatively low.

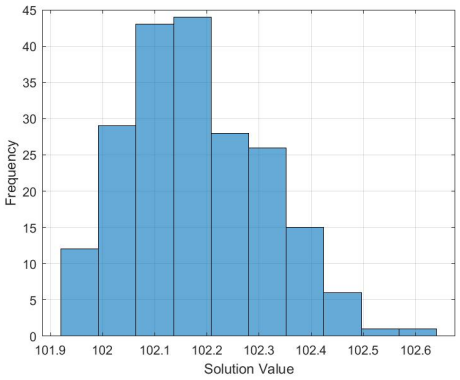


Figure 7: This figure illustrates the solution value after subsequently executing simulated annealing and path generation.

Simulated annealing, as well as path generation, adjusts the initial schedule of either the first x_1 or the second day x_2. Considering all neighbourhoods of simulated annealing, approximately 60% of the applied adjustments are changes to the schedule x_1, whereas only 40% of the changes are applied to the succeeding day. Even more remarkable is the difference for path generation: 85% of the changes are applied to the schedule of the first day x_1. Because of the limited solution space, path generation can only implement adjustments between the last arrival and the first departure. In general, the first day is more flexible, because the time between the last rush hour and the end of the day is larger compared to the time between the start of the following day and the morning rush hour. The ratio of implementing changes to x_1 or x_2 is nearly identical for every instance. It is also remarkable to mention that adjustments to the composition of night trains are barely used to dissolve unbalances.

The main factor for the difference between the solution approaches and the upper bound is the reduction of the number of deadheading trips. Due to the penalty factor γ, the attractiveness of choosing an empty movement is lower. Reducing the number of deadheading trips comes mainly at the cost of an increase in the number of carriages kilometres and the number of shunting movements in the initial schedule. The number of seat shortages barely differs between the initial schedules, the optimal schedule or the introduced solution approaches.

Finally, let us discuss how often the deadheading movement is used to dissolve unbalances. The initial number of unbalances are depicted in Table 1. The upper bound solves the unbalances by only using deadheading movements. Simulated annealing tries to avoid deadheading movements by changing the inventory levels. After applying simulated annealing, the number of remaining unbalances is approximately 60 - 80% compared to the initial number of unbalances. The reduction in the number of unbalances for path generation is similar: the number of remaining deadheading movements fluctuates between 50 and 70%. Here, approximately 40% of these unbalances were dissolved by only a deadheading movement, whereas the remaining 60% were solved by a combination of a deadheading movement and an adjustment to the composition of one or multiple trips.

In the overall heuristic approach, the number of deadheading movement is approximately 35 - 60% compared to the initial number of unbalances. On the other hand, the optimal solution only consists of approximately 10 - 25% unbalances with an empty movement. Although the heuristic consists of more deadheading movement travelling a short distance (less than 50km), the total number of deadheading kilometres is still considerably higher for the heuristic compared to the optimal solution.

7.4 Sensitivity Analysis

Now, we investigate whether modifications to the parameters influence the results of the proposed methods. Here, the carriage kilometres and, especially, the seat-shortage kilometres are not noteworthy. These factors are crucial for the rolling stock circulation problem, but are of less importance for rolling stock rebalancing. As noted in the previous section, the number of seat-shortage kilometres does not seem to increase or decrease for any of the introduced methods. Furthermore, coupling a train unit to a trip as well as executing an empty movement both lead to an increase in the number of carriage kilometres.

However, the parameter of the shunting movements is more meaningful for this problem. Increasing this factor prevents the usage of shunting movements and, therefore, develops a less flexible rolling stock schedule. Then, (de)coupling is relatively expensive in comparison to a straightforward deadheading trip. As a result, more deadheading movements are included in the final schedule. In Table 5, the results of every solution approach for different weight factors of shunting movements can be found. Here, LB could denote two different values. Firstly, LB could be equal to the optimal solution value, if the gap of 0.01% is reached within 30 minutes hour. An asterisk next to the value denotes that this number is the optimal solution value. Otherwise, similar to the previous tables, LB does actually represent the lower bound.

Table 5: Comparison for different values of the parameter for (de)coupling movements.

	$C_{SHM} = 1$					$C_{SHM} = 100$				
Instance	**LB**	**Heur**	**SA**	**Unb.**	**UB**	**LB**	**Heur.**	**SA**	**Unb.**	**UB**
1	100*	100.1	100.1	100.1	101.0	100*	100.0	100.0	100.5	100.9
2	100*	102.0	103.6	108.4	108.9	100*	100.7	103.0	104.4	105.3
3	100*	100.7	101.3	102.9	102.9	100*	103.5	103.8	105.8	105.8
4	100*	101.0	101.2	101.2	101.4	100*	100.3	100.4	100.6	100.7
5	100*	100.3	100.8	100.7	101.0	100*	100.3	100.4	100.4	100.5
6	100	106.9	109.3	108.2	110.7	100	104.2	105.6	105.6	106.8
7	100	102.1	103.7	102.5	103.9	100	101.9	103.1	102.1	103.1
8	100	100.8	101.4	100.9	101.6	100	101.0	102.2	101.2	101.7
9	100	106.9	108.5	109.4	109.6	100	105.8	107.9	106.9	108.7
10	100	100.4	100.8	101.4	101.6	100	101.1	101.4	101.3	101.6

Each rolling stock schedule is different for various settings of the parameter C_{SHM}. Therefore, the gap between the lower and upper bound differs for each unique value of the parameter C_{SHM}. Let us take a look at the results of the path generation approach. In most cases, the relative gap of *Unb.* to the lower bound is higher whenever $C_{SHM} = 1$ in comparison to higher values of C_{SHM}. However, this does not mean that this method performs better for higher values of C_{SHM}. On the contrary, the relative percentage gap between *Unb.* and *UB* is higher for lower values of C_{SHM}. In other words, the number of alternatives for an expensive deadheading movement increases, because a shunting movement is relatively cheap.

Equivalently for the other methods, the percentage gap to the lower bound is in general higher when the value of C_{SHM} decreases, but meanwhile, the relative gap to the upper bound becomes larger. Nevertheless, for both parameter values, the solution approaches still produce decent results. Especially simulated annealing does not seem to be influenced by an adjustment of C_{SHM}. For a lower value of C_{SHM}, the number of iterations of simulated annealing is higher and the number of remaining deadheading trips is eventually lower.

Just as the parameter of the shunting movements, the penalty factor of deadheading movements is noteworthy. Decreasing the penalty factor encourages the usage of empty movements, while increasing γ prevents us from using these trips. Table 6 illustrates the solution values for different penalty factors of deadheading movements.

Table 6: Comparison for different values of the penalty factor for the empty movements.

	$\gamma = 1$					$\gamma = 10$				
Instance	**LB**	**Heur**	**SA**	**Unb.**	**UB**	**LB**	**Heur.**	**SA**	**Unb.**	**UB**
1	100*	100.00	100.00	100.00	100.47	100*	101.7	104.6	104.8	112.1
2	100*	100.13	100.19	100.54	100.54	100*	103.3	114.4	155.5	169.8
3	100*	100.20	100.32	100.39	100.39	100*	101.4	102.0	114.3	114.3
4	100*	100.05	100.08	100.08	100.08	100*	101.3	102.1	102.6	102.9
5	100*	100.11	100.11	100.11	100.11	100*	101.1	101.6	101.7	102.0
6	100	101.00	101.00	101.01	101.01	100	121.6	126.1	125.0	135.3
7	100	100.26	100.26	100.29	100.29	100	104.7	108.9	106.4	111.3
8	100	100.14	100.14	100.16	100.16	100	102.1	104.3	102.4	105.5
9	100	100.91	100.93	100.98	100.98	100	119.8	128.7	128.4	129.0
10	100	100.13	100.14	100.15	100.16	100	102.2	103.1	103.2	103.8

Clearly, the penalty factor γ highly influences the gap between the lower and upper bound. Almost all instances are within 1% of the lower bound, whenever $\gamma = 1$. The costs of a deadheading trip between station A and B for $\gamma = 1$ is just as expensive as the costs of including an additional train unit to a trip between station A and B, provided the initial composition does not have any seat shortages. Moreover, if the passenger trip between station A and B requires at least one shunting movement, the deadheading movement is even a better, least-cost alternative. Even in the optimal solution, the number of used deadheading trips is approximately 95% in comparison to the number of initial unbalances.

Path generation hardly seems to find any improvements compared to the initial schedule. Due to its limited solution space, it can hardly find any better alternatives than a deadheading trip. Simulated annealing mainly cancels and subsequently adds shunting movements, such that it can solve some unbalances. Nevertheless, hardly any unbalances are dissolved.

On the other hand, when the factor for deadheading trips increases, the gap between the lower and upper bound increases substantially. *Unb.* provides decent results in comparison to the situation where $\gamma = 1$. Especially for instances representing weekdays, the time-dependent graphs can easily find paths cheaper than the deadheading movement. Similarly, but less obvious, simulated annealing tries to avoid unbalances by changing the start-of-day and end-of-day inventories. It is noteworthy to mention that the number of iterations of simulated annealing increases as γ increases. The strength of the overall heuristic follows from the relatively low gap to the lower bound. For example, take a look at instance 2 and 7. The self-contained methods *SA* and *Unb.* still have a significant gap to the lower bound, but when both methods are used in *Heur.*, it leads to a significant reduction of the objective value.

8 Conclusion

The main focus of this thesis is to tackle the rolling stock rebalancing problem. This problem arises if the rolling stock circulation problem of multiple consecutive days is determined for each day independently. Although technically speaking, it is possible to determine the rolling stock schedule of multiple consecutive days without considering the days separately, the running time seems to explode exponentially. In practice at NS, the schedules are calculated for each day separately. Then, the schedule is in unbalance. The goal of the rolling stock rebalancing problem is to dissolve all arisen unbalances. Here, we assume that the (more expensive) deadheading movements are allowed during the night.

In this thesis, multiple solution approaches are proposed to tackle rolling stock rebalancing. First of all, the sequential solution approach re-uses the composition model of Fioole et al. (2006). The idea is to re-optimise a narrow, predefined time window such that the unbalances are solved. The results clearly show a trade-off between running time and solution quality. As the length of the time window expands, the running time also seems to grow exponentially. Nonetheless, the solution value also gradually decreases as the time window increases. When the time window is relatively small, the solution value decreases more rapidly than for wider time windows. In conclusion, the method performs decently for smaller to medium-sized time windows, since there is a clear trade-off between running time and the quality of the solution.

Next, the heuristic approach consists of two distinctive methods, which could be merged into an overall solution approach. A time-dependent graph is used to represent all possibilities a train unit can traverse from one station to another station. Then, an approach based on column generation is presented which repeatedly generates (the shortest) paths of time-dependent graphs. The results show that this method is relatively effective and has a relatively low computation time. However, an unbalance can only be dissolved between the last arrival of the first day and the first departure of the succeeding day. For this reason, the disadvantage of this method is its relatively small solution space.

Simulated annealing is used to diversify the solution space. The idea of simulated annealing is to implement adjustments to the initial schedule such that the number of (weighted) unbalances decreases. The neighbourhoods, which change the initial schedule, are randomly chosen during simulated annealing. The results show that this approach does not necessarily dominate the path generation approach or vice versa. Nevertheless, the solution space is larger and the quality of the solution could possibly increase if even more neighbourhoods are introduced.

The overall solution approach uses path generation and simulated annealing in an iterative procedure. Because simulated annealing tries to decrease the number of (weighted) unbalances by randomly changing the inventory levels, the remaining unbalances are different for each iteration. Therefore, path generation repeatedly solves a different set of remaining unbalances and its obtained solution is different for each iteration. The analysis of the results displays that the proposed heuristic produces decent results. It clearly illustrates the effectiveness of the combination of path generation and simulated annealing. Nevertheless, the gap between the heuristic and the lower bound is still to some extent significant. On the other hand, the computation time is relatively low compared to the exponentially increasing running time for determining the optimal solution.

Furthermore, the quality of the methods remains comparable if the start-of-day inventory of the first day and the end-of-day inventory of the second day are fixed. Although the solution values slightly increase, the solution approaches could still produce decent results. Besides that, the sensitivity analysis reveals that adjusting the parameters does not lead to a significant quality loss of the proposed solution approaches. Although the schedules are highly dependent on these parameter settings, the relative gap between the methods and the lower bound remains proportionate. Nevertheless, setting the parameter value of deadheading trips γ to 1 negatively influences the results, but this specific setting does also not represent the reality.

In conclusion, the proposed solution methods produce decent results, where especially the running time remains low. The combination of the random nature of simulated annealing and the speed of path generation leads to a significant decrease of deadheading trips. Adjusting the parameters clearly showed some interesting insights into the methods. The perfect environment for the methods is a higher (and realistic) factor γ for deadheading trips and rolling stock which is relatively flexible in terms of shunting possibilities.

9 Discussion

Although the proposed solution approaches produced decent solutions, there is also room for improvement. The gap between the proposed solution approaches and the lower bound or optimal solution is still non-negligible. The solution space of the methods of Sections 6.3 and 6.4 is still limited compared to the exact mathematical formulation of Fioole et al. (2006). Adding more (advanced) neighbourhoods could possibly decrease the solution value, but probably at the cost of additional running time. Similarly, considering more, or even all feasible paths for the master problem of Section 6.3, could lead to a cost-reduction. But also, in this case, the room for improvement is minimal due to the relatively limited solution space.

The assumptions stated in this thesis seem realistic. The main assumption, that a deadheading trip between two arbitrarily stations at night is always possible, also applies in practice. Although, in reality, deadheading movements are executed as soon as possible and not specifically at night. The required staff to execute an empty movement is in general cheaper in the evening than during the night. Therefore, the costs of a deadheading trip could depend on whether the movement is executed at 19:00 or at 01:00. Although this discrepancy in costs is not used in this thesis, it can easily be implemented by changing the definition of the costs of an empty movement $C^d_{s_1,s_2,m}$.

An idea for further research of the rolling stock rebalancing problem would be to use a decomposition method. A decomposition method such as Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) could be used to take advantage of the special structure of this problem. Determining the rolling stock schedule of a day can be seen as a subproblem. Then, equalling the end-of-day and start-of-day inventory of consecutive days could be interpreted as the connecting or coupling constraints.

An alternative to the consideration of rolling stock rebalancing would be to examine a more integral method. Unbalances do not only appear due to the independently generated roster, but also as a result of the provided timetable. By way of explanation, let us consider a station A. Suppose that n trips leave station A on a certain day before the first train arrives at station A. Hence, at least n units are required at station A to cover all leaving trips. But, also on the evening before, more trips leave station A than the number of trips that arrive at station A and/or the decoupling movements at station A are limited. Accordingly, station A is in unbalance by definition, even though the rolling stock schedule is still unknown. Therefore, it would be interesting to consider an integrated solution approach of timetabling and rolling stock circulation, that checks and subsequently eliminates these avoidable unbalances.

Lastly, one could also question whether a priority for the rolling stock rebalancing problem would be essential, especially when one considers the strategic or tactical planning horizon. The relative gap between the optimal, balanced solution and the proposed solution approaches is only a few percentage points. Obviously, a cost-reduction on a repeating, weekly roster eventually leads to a substantial decrease in costs. But, in practice, the initial, well-thought rolling stock schedule cannot always be executed as planned due to (daily) maintenance planning, delays and/or disruptions. As a result, the end-of-day inventory is completely different compared to the initial plan. The initial solution of dissolving a specific unbalance cannot be used. Hence, the dynamics elements of short-term planning heavily influence the actual execution of the initial schedule.

References

Bellman, R. (1958). On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90.

Borndörfer, R., Reuther, M., Schlechte, T., Waas, K., and Weider, S. (2016). Integrated optimization of rolling stock rotations for intercity railways. *Transportation Science*, 50(3):863–877.

Budai, G., Maróti, G., Dekker, R., Huisman, D., and Kroon, L. (2010). Rescheduling in passenger railways: the rolling stock rebalancing problem. *Journal of scheduling*, 13(3):281–297.

Cacchiani, V., Caprara, A., and Toth, P. (2010). Solving a real-world train-unit assignment problem. *Mathematical programming*, 124(1-2):207–231.

Cadarso, L. and Marín, Á. (2012). Integration of timetable planning and rolling stock in rapid transit networks. *Annals of Operations Research*, 199(1):113–135.

Dantzig, G. B. and Wolfe, P. (1960). Decomposition principle for linear programs. *Operations research*, 8(1):101–111.

Dijkstra, E. W. et al. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Farina, F., Huisman, D., Roberti, R., and Azadeh, S. S. (2018). Rebalancing rolling stock by scheduling deadhead trains.

Fioole, P.-J., Kroon, L., Maróti, G., and Schrijver, A. (2006). A rolling stock circulation model for combining and splitting of passenger trains. *European Journal of Operational Research*, 174(2):1281–1297.

Floyd, R. W. (1962). Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345.

Haahr, J. and Lusby, R. M. (2017). Integrating rolling stock scheduling with train unit shunting. *European Journal of Operational Research*, 259(2):452–468.

Haahr, J. T., Lusby, R. M., Larsen, J., and Pisinger, D. (2014). A branch-and-price framework for railway rolling stock rescheduling during disruptions.

Haahr, J. T., Wagenaar, J. C., Veelenturf, L. P., and Kroon, L. G. (2016). A comparison of two exact methods for passenger railway rolling stock (re) scheduling. *Transportation Research Part E: Logistics and Transportation Review*, 91:15–32.

Hershberger, J., Maxel, M., and Suri, S. (2007). Finding the k shortest simple paths: A new algorithm and its implementation. *ACM Transactions on Algorithms (TALG)*, 3(4):45–es.

Hoogervorst, R., Dollevoet, T., Maróti, G., and Huisman, D. (2019). A variable neighborhood search heuristic for rolling stock rescheduling. Technical report.

Lin, Z. and Kwan, R. S. (2014). A two-phase approach for real-world train unit scheduling. *Public Transport*, 6(1-2):35–65.

Lin, Z. and Kwan, R. S. (2016). A branch-and-price approach for solving the train unit scheduling problem. *Transportation Research Part B: Methodological*, 94:97–120.

Lusby, R. M., Haahr, J. T., Larsen, J., and Pisinger, D. (2017). A branch-and-price algorithm for railway rolling stock rescheduling. *Transportation Research Part B: Methodological*, 99:228–250.

Marchetti-Spaccamela, A., Nanni, U., and Rohnert, H. (1996). Maintaining a topological order under edge insertions. *Information Processing Letters*, 59(1):53–58.

Maróti, G. (2006). Operations research models for railway rolling stock planning.

Nielsen, L. K. (2011). *Rolling Stock Rescheduling in Passenger Railways: Applications in short-term planning and in disruption management.* Number EPS-2011-224-LIS.

Nielsen, L. K., Kroon, L., and Maróti, G. (2012). A rolling horizon approach for disruption management of railway rolling stock. *European Journal of Operational Research*, 220(2):496–509.

Schöbel, A. (2017). An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation. *Transportation Research Part C: Emerging Technologies*, 74:348–365.

Tréfond, S., Billionnet, A., Elloumi, S., Djellab, H., and Guyon, O. (2017). Optimization and simulation for robust railway rolling-stock planning. *Journal of Rail Transport Planning & Management*, 7(1-2):33–49.

UIC (2015). Uic railway statistics 2015.

Wang, Y., Gao, Y., Yu, X., Hansen, I. A., and Miao, J. (2019). Optimization models for high-speed train unit routing problems. *Computers & Industrial Engineering*, 127:1273–1281.

Zhong, Q., Lusby, R. M., Larsen, J., Zhang, Y., and Peng, Q. (2019). Rolling stock scheduling with maintenance requirements at the chinese high-speed railway. *Transportation Research Part B: Methodological*, 126:24–44.