Journal Pre-proof

A Variable Neighborhood Search Heuristic for Rolling Stock Rescheduling

Rowan Hoogervorst, Twan Dollevoet, Gábor Maróti, Dennis Huisman

PII: S2192-4376(21)00004-2

DOI: https://doi.org/10.1016/j.ejtl.2021.100032

Reference: EJTL 100032

To appear in: *EURO Journal on Transportation and Logistics*

Received Date: 28 January 2020

Revised Date: 20 December 2020

Accepted Date: 22 January 2021

Please cite this article as: Hoogervorst, R., Dollevoet, T., Maróti, G., Huisman, D., A Variable Neighborhood Search Heuristic for Rolling Stock Rescheduling, *EURO Journal on Transportation and Logistics*, https://doi.org/10.1016/j.ejtl.2021.100032.

A Variable Neighborhood Search Heuristic for Rolling Stock Rescheduling

Rowan Hoogervorst[1], Twan Dollevoet[1], Gábor Maróti[2,3], and Dennis Huisman[1,3]

[1]Econometric Institute and ECOPT, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands
[2]VU University Amsterdam, 1081 HV Amsterdam, The Netherlands
[3]Process quality and Innovation, Netherlands Railways, 3500 HA Utrecht, The Netherlands

Abstract

We present a Variable Neighborhood Search heuristic for the rolling stock rescheduling problem. Rolling stock rescheduling is needed when a disruption leads to cancellations in the timetable. In rolling stock rescheduling, one must then assign duties, i.e., sequences of trips, to the available train units in such a way that both passenger comfort and operational performance are taken into account. For our heuristic, we introduce three neighborhoods, which focus on swapping duties between train units, on improving the individual duties and on changing the shunting that occurs between trips, respectively. These neighborhoods are used for both a Variable Neighborhood Descent local search procedure and for perturbing the current solution in order to escape from local optima. Moreover, we show that the heuristic can be extended to the setting of flexible rolling stock turnings at ending stations by introducing a fourth neighborhood. We apply our heuristic to instances of Netherlands Railways (NS). The results show that the heuristic is able to find high-quality solutions within one minute of solving time. This allows rolling stock dispatchers to use our heuristic in real-time rescheduling.

Keywords— Disruption Management, Rolling Stock Rescheduling, Variable Neighborhood Search

1 Introduction

There is a significant need for decision support for rolling stock rescheduling at train operating companies. On the one hand, rolling stock rescheduling has a large impact on the passenger satisfaction, as passengers have to stand or even wait for the next train if not enough rolling stock is available to operate a trip. On the other hand, having an efficient rescheduling process in place also limits the number of train units that are needed as a buffer to deal with disruptions, which significantly lowers the capital costs of a train operator.

In this paper, we focus on the real-time rescheduling of rolling stock after a disruption leads to the cancellation of trips in the timetable. This, e.g., happens when a section of railway infrastructure becomes blocked due to a mechanical failure, which leads to the cancellation of all trips using this section of infrastructure for the duration of the disruption. Alternatively, this may happen when a mechanical failure on a train unit causes this train unit to be unable to continue the trips it is currently operating. An average of 16 disruptions was reported per day in 2019 in the passenger information systems for the Dutch railway network.[1]

As a result of the timetable adjustments due to a disruption, the train units that were planned to operate these canceled trips will be unable to reach the station they were supposed to arrive at after these trips. As these train units may be planned to operate other trips starting at this station, this is likely to lead to further cancellations if no rescheduling is performed. It is these knock-on effects of the disruption on the rolling stock assignment that we focus on in this paper, where we try to minimize any further cancellations and try to ensure that there is a good balance between the number of available seats and the passenger demand. This is done while trying to minimize the changes that are made compared to the original rolling stock assignment.

Existing literature for rolling stock rescheduling has mostly focused on exact solution methods, of which many were originally developed for the rolling stock scheduling setting. State-of-the-art models include those by Fioole et al. (2006), Borndörfer et al. (2016) and Lusby et al. (2017). While these exact models have been extremely successful in solving a large number of variations of the rolling stock rescheduling problem, they also have some fundamental difficulties when used in practice for real-time rolling stock rescheduling.

[1] `https://www.rijdendetreinen.nl/en/statistics/2019`

Most importantly, these exact methods generally consider a simplified version of the rolling stock rescheduling problem that is faced by dispatchers. For example, the model of Fioole et al. (2006) considers all train units that are of the same rolling stock type as fully interchangeable. While this is a realistic assumption in rolling stock scheduling, where the exact train unit is still likely to change between the moment of planning and the day of operation, choosing the correct train unit is often important in real-time rescheduling. This is, e.g., the case when a train unit has a small mechanical defect which limits how the train unit can be used. A broken windshield wiper may, for example, cause one of the two cabins of a train unit to become unusable for a driver, which implies that this cabin cannot be at the front of the train.

Numerous papers have extended the existing models for issues encountered in rescheduling. Examples include the incorporation of maintenance constraints by Wagenaar, Kroon, and Schmidt (2017), the incorporation of train delays by Hoogervorst et al. (2020) and the incorporation of deadheading by Wagenaar, Kroon, and Fragkos (2017). Moreover, Nielsen (2011) considers a setting where the turnings between trips at ending stations can be chosen freely, which is referred to as flexible turning. A disadvantage of these approaches is that the time needed to solve these models rises quickly when expanding the model. As a result, exact solution methods often struggle to solve instances when including all, or many, of the details that are considered by rolling stock dispatchers.

Based on the above observation, the main contribution of this paper is to propose a Variable Neighborhood Search heuristic for the rolling stock rescheduling problem. For this heuristic, we introduce three local search neighborhoods for the rolling stock rescheduling problem, which can be applied to a wide variety of rolling stock rescheduling settings. In addition, to show that the heuristic can easily be extended to more involved rescheduling contexts, especially ones that are hard for exact methods, we show how a fourth neighborhood can be added to deal with the flexible turning setting of Nielsen (2011). We test the heuristic on instances of Netherlands Railways (NS), where we evaluate the quality of the provided solutions by comparing them to an exact solution method.

The remainder of the paper is organized as follows. In Section 2, we define the rolling stock rescheduling problem. In Section 3, we discuss the existing literature on rolling stock rescheduling. In Section 4, we introduce our Variable Neighborhood Search algorithm and the three neighborhoods that are considered in the algorithm. We extend the heuristic to the rolling stock rescheduling setting with flexible turning in Section 5. We then apply the heuristic to instances of NS in Section 6. Lastly, we conclude the paper in Section 7.

2 The Rolling Stock Rescheduling Problem

In rolling stock rescheduling, we assign rolling stock to the set of trips $\mathcal{T}$ in the timetable. Each trip is characterized by its departure station, arrival station, departure time and arrival time. Let $\mathcal{S}$ be the set of all stations. We will assume that the timetable has already been updated for the initial disruption, implying that all cancellations that follow directly from the disruption have been incorporated into the timetable.

Most train operators use a heterogeneous fleet of train units to operate the trips. Let $\mathcal{R}$ be the set of available train unit types. In this paper, we focus on self-propelled train units, i.e., no locomotive is needed to pull them. It is often possible to couple train units of compatible types together to form *compositions*, which are sequences of train unit types in $\mathcal{R}$. Let $\mathcal{P}$ indicate the set of all possible compositions. An example of a composition is given in Figure 1, where three train units of the ICM family of train units are coupled together to form a composition.

Figure 1: An example of a composition consisting of two ICM-III train units at the front and back of the composition and one ICM-IV train unit in the middle.

As we consider a rescheduling setting, the size of the rolling stock fleet is fixed. In particular, let $\iota^0_{r,s}$ indicate the number of train units of type $r \in \mathcal{R}$ that start at station $s \in \mathcal{S}$. Moreover, let $\iota^\infty_{r,s}$ denote the number of train units of type $r \in \mathcal{R}$ that are supposed to end at station $s \in \mathcal{S}$. Such a target number of train units to end at each station is imposed to facilitate the start of the timetable on the next day.

We first assume that fixed rolling stock connections are defined between incoming and outgoing trips at a station. Let $\mathcal{C}$ be the set of these rolling stock *transitions*. Each transition $c \in \mathcal{C}$ is defined by the sets $\mathcal{T}^-_c$ and $\mathcal{T}^+_c$, which indicate the incoming trips and outgoing trips in this transition, respectively. Note that a transition normally links one incoming to one outgoing trip, implying that both sets contain a single trip, and that one of these sets can be empty in case all rolling stock comes from or is moved to the shunting yard. In Section 5, we consider the setting with flexible turning, in which the rolling stock connections at terminal stations can be optimized (Nielsen 2011). In that case, determining the set of transitions $\mathcal{C}$ is part of the optimization problem.

The compositions that can be operated on a given trip are limited by infrastructure and operational constraints, such as the maximum platform length at the stations. Let $\mathcal{P}_t \subseteq \mathcal{P}$ denote the compositions that can be used to operate trip $t \in \mathcal{T}$. The way in which compositions can be changed at transitions is also limited. For example, coupling and uncoupling train units is often only possible on one side of the composition due to the station layout. Let $\mathcal{Q}_c$ be the set of possible composition changes for transition $c \in \mathcal{C}$. Each composition change is characterized by the compositions $q(t) \in \mathcal{P}_t$ for each $t \in \mathcal{T}_c^- \cup \mathcal{T}_c^+$, i.e., by the compositions on all incoming and outgoing trips of the transition. Then, let $\mathcal{Q} = \bigcup_{c \in \mathcal{C}} \mathcal{Q}_c$ denote the set of all possible composition changes.

In rolling stock rescheduling our goal is now to assign *duties* to the available train units, where each duty describes the trips that are operated by that unit during the planning horizon and the position it takes in the composition of each of these trips. Let $\mathcal{D}$ be the set of duties, which is often referred to as a rolling stock *circulation*. Note that $\mathcal{D}$ defines both an assignment $\mathcal{T} \to \mathcal{P}$ of compositions to the trips and an assignment $\mathcal{C} \to \mathcal{Q}$ of composition changes to the transitions. The circulation $\mathcal{D}$ is only feasible if the assignment $\mathcal{T} \to \mathcal{P}$ assigns to every trip t a composition in $\mathcal{P}_t$ and the assignment $\mathcal{C} \to \mathcal{Q}$ assigns to every transition c a composition change in $\mathcal{Q}_c$.

An example of a circulation is given in Figure 2. In this example, we consider a timetable that consists of twelve trips $(t_1, \ldots, t_{12})$ between three stations (*Rtd*, *Gd*, *Ut*). The timetable is shown through a time-space diagram, where an arc connects two stations if there is a trip between them. Moreover, the figure shows a circulation consisting of four duties, two for train units of the red (dashed) rolling stock type and two for train units of the blue (solid) rolling stock type. Note that the circulation implies a composition consisting of two train units on trips $t_1, \ldots, t_4$ and compositions of a single train unit on the other trips. Table 1 describes three transitions from the example. Transition c_3 is a so-called turning, where a train arrives at a terminal stations and returns to the origin station it departed from. During such a transition, no passengers are present in the train and the composition typically switches order.

To judge the quality of a circulation, we assign costs to the compositions and composition changes in the circulation, and to any deviation from the target number of train units to end at each station at the end of the planning horizon. Our most important target is to prevent any cancellations,

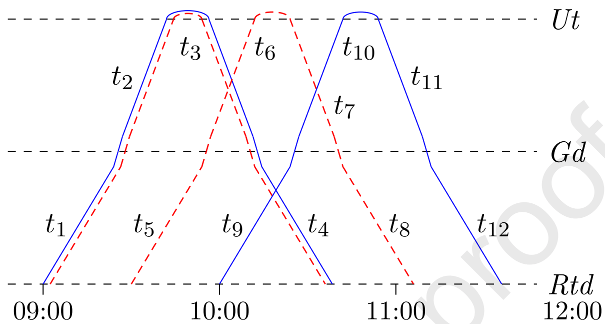


Figure 2: An example of a circulation for 4 train units, two of the blue (solid) and two of the red (dashed) type.

Table 1: Three examples of transitions depicted in Figure 2. Here, B and R correspond to a blue (solid) and red (dashed) train unit, respectively.

Transition c	$\mathcal{T}_c^-$	$\mathcal{T}_c^+$	Composition change q
c_1	$\emptyset$	$\{t_1\}$	$q(t_1) = (B, R)$
c_2	$\{t_1\}$	$\{t_2\}$	$q(t_1) = (B, R)$, $q(t_2) = (B, R)$
c_3	$\{t_2\}$	$\{t_3\}$	$q(t_2) = (B, R)$, $q(t_3) = (R, B)$

as cancellations have a large impact on the journey of the passengers. A cancellation is modeled as assigning an empty composition to a trip, which is penalized with a high cost. Moreover, we try to ensure that the chosen compositions offer enough seat capacity while limiting the number of kilometers that the train units make. For the composition changes, we like to avoid any changes in shunting activity. For example, introducing additional coupling of train units may require changes to the duties of the crews at the stations and is thus penalized. Similarly, we would like to minimize any deviations from the target number of train units that are supposed to end at each station at the end of the day, as additional deadheading trips during the night might be necessary to alleviate any off-balances.

The rolling stock rescheduling problem is now formally defined as follows. Given an updated set of trips $\mathcal{T}$ and transitions $\mathcal{C}$, define a set of duties $\mathcal{D}$ with one duty for every train unit, such that the implied composition on trip $t \in \mathcal{T}$ is in $\mathcal{P}_t$ and the implied composition change for transition $c \in \mathcal{C}$ is in $\mathcal{Q}_c$. The aim is to minimize the costs for the assigned compositions and composition changes, and for any deviations from the target number of train units of each type that end at each station.

In Section 5, we extend the problem with flexible turning. In this setting, the set of transitions must be determined as part of the rolling stock rescheduling problem.

3 Literature Review

The rescheduling of rolling stock is one of the typical steps in the rescheduling process of a passenger railway operator. An overview of the rescheduling process and the steps taken in it is given by Cacchiani et al. (2014). In particular, note that the timetable is normally rescheduled before the rolling stock is rescheduled and that the crew is rescheduled afterward. While some papers have focused on further integrating some of these steps, see, e.g., Veelenturf et al. (2016), it has been shown that this sequential process performs well in practice by Dollevoet et al. (2017).

A large body of literature is available on the rolling stock rescheduling problem, where variations in the problem setting have led to different models. In particular, such model variations are often a result of the different infrastructural and operational contexts encountered by the various railway operators. In this paper, we specifically focus on passenger railway operators and on rolling stock that consists of train units, is electrically powered and can drive in both directions, implying that no locomotives have to be considered in the rescheduling process.

Numerous exact solution approaches have been proposed for the rolling stock rescheduling problem, where most of these models were originally developed for rolling stock scheduling. A multi-commodity flow based model for rolling stock scheduling has been proposed by Fioole et al. (2006), which was extended by Nielsen, Kroon, and Maróti (2012) to the rescheduling setting. An important assumption in this model is that all train units of the same type are considered interchangeable, allowing an aggregation of the train units that are stored at the shunting yard of a station at any moment in time.

Alternatively, Borndörfer et al. (2016) solve the rolling stock scheduling problem by finding a flow in an appropriate hypergraph. Recently, a rescheduling approach based on this model has been used by Borndörfer, Grimm, and Schlechte (2019). Other models for rolling stock rescheduling using a hypergraph include those by Borndörfer et al. (2014) and Borndörfer et al. (2017). Unlike Fioole et al. (2006), this model allows to track individual train units, which is used to enforce maintenance constraints based on the distance driven by a train unit. Moreover, while the model of Fioole et al. (2006) can be solved by means of a commercial MIP solver, a column generation approach is needed to solve this model due to the large number of hyperarcs. Both the composition model and the hypergraph-model are discussed by Reuther (2017).

Instead of making use of a flow based formulation, Cacchiani, Caprara, and Toth (2010) and Lusby et al. (2017) generate duties for the individual train units. Due to the large number of possible duties, they, like Borndörfer et al. (2016), also use column generation to solve the model. A different column generation based approach has been considered by Peeters and Kroon (2008). Instead of generating columns that represent duties, they generate columns that correspond to paths in the transition graph, which describes the compositions on the trips and how these compositions can be changed at transitions between trips.

While the above models can all be applied or have been applied to a rolling stock rescheduling context, they are mostly aimed for rescheduling the rolling stock before the day of operation. Numerous papers have extended the above models to include additional details of the real-time rescheduling process. Wagenaar, Kroon, and Fragkos (2017) consider the possibility of adding deadheading trips and take into account the changes to the passenger demand after a disruption. Similarly, Wagenaar, Kroon, and Schmidt (2017) consider the effect of maintenance appointments, in which train units need to be present at a station at a specified time for maintenance. The minimization of passenger delays by means of rescheduling the rolling stock assignment is considered by Hoogervorst et al. (2020). When considering all of these papers, we see that the inclusion of additional details generally leads to an increase in the computation time.

Concurrently to the successful introduction of exact methods, also heuristics have been proposed to solve the rolling stock scheduling and rescheduling problem. Cacchiani, Caprara, and Toth (2019) propose a heuristic for rolling stock scheduling that extends an optimal rolling stock assignment for the peak hours to a rolling stock assignment for the complete day. They apply the heuristic to instances of a railway operator in Northern Italy. Unlike our heuristic, their heuristic does not make use of local search to improve a found solution.

Another matheuristic, i.e., a heuristic based on mathematical programming, for the rolling stock scheduling problem is that of Cacchiani, Caprara, and Toth (2013). In this heuristic, the authors apply Lagrangian relaxation based on an arc formulation of the problem to obtain lower bounds. Moreover, a Lagrangian heuristic is used to obtain feasible solutions, in which a constructive procedure transforms an infeasible solution into a feasible one and local search improves the found solution. Like our paper, also Cacchiani, Caprara, and Toth (2013) use neighborhoods to find improved solutions in their Lagrangian heuristic. However, the problem considered is significantly different, as Cacchiani, Caprara, and Toth (2013) do not consider the order of train units within a composition and do not consider fixed transitions like we do.

A paper that is closely related to ours is the one of Budai et al. (2010), although they focus on a sub-problem of the rolling stock rescheduling problem. In particular, they try to solve any off-balances that are present in the number of train units that end at each station when compared to the planned number of train units to end at each station, a problem they call the Rolling Stock Rebalancing Problem. To solve the off-balances, they exploit the network flow properties of the problem. We will employ a similar idea in two of the presented neighborhoods to find improvements to the current circulation.

4 VNS Heuristic for Rolling Stock Rescheduling

We propose a Variable Neighborhood Search (VNS) heuristic to solve the Rolling Stock Rescheduling Problem. VNS is a local search based metaheuristic that has been proposed by Mladenović and Hansen (1997). The main idea in VNS is to iteratively explore multiple neighborhoods, both in terms of local search and in terms of perturbation, where one switches to a larger sized neighborhood if a smaller neighborhood does not yield an improvement to the current solution.

In our VNS heuristic, we consider three different neighborhoods. First, we consider a neighborhood that corresponds to swapping duties between train units. Second, we consider a neighborhood that improves the assignment for a single train unit type. Third, we consider a neighborhood that corresponds to adjusting the composition changes at transitions. In the remainder of this section, we introduce these three neighborhoods and the VNS heuristic that explores these neighborhoods.

4.1 Two-Opt Duty Neighborhood

In the Two-Opt Duty neighborhood, we focus on improving the assignment of the different rolling stock types to the trips. We do so by swapping the remaining parts of the duties of two train units, of different rolling stock type, when these train units meet at a station. As a result of this swap, we change the assignment of compositions to trips and that of composition changes to the transitions.

An example of a move in which two duties are exchanged is given in Figure 3, where the duties of the highlighted red (dashed) and blue (solid) train units are switched. This swap is possible, as both train units are present at station *Rtd* prior to the departure time of trip t_5. Note that the swap alters the compositions that are used on respectively trips $t_5, \ldots, t_{12}$ and the composition changes that are used on those transitions that precede and succeed these trips. This would, for example, be beneficial if altering these compositions leads to a better matching of capacity with passenger demand on these trips. Specifically, if more passengers are expected on trips $t_5, \ldots, t_8$ than on trips $t_9, \ldots, t_{12}$ and blue (solid) train units offer more seats than red (dashed) units, then the swap improves the objective value.

We can efficiently find the possible moves by noting that the remaining parts of the duties of two train units can only be exchanged if both train units are present at the same station at some moment in time. This follows from the fact that shunting is only executed to couple and uncouple train units. We can then enumerate all rolling stock duties that are available for coupling at some transition, which is the case if the train unit is present at the station at the moment that this coupling action is started. We can then exchange two duties, of different rolling stock types, that are available at the same transition.

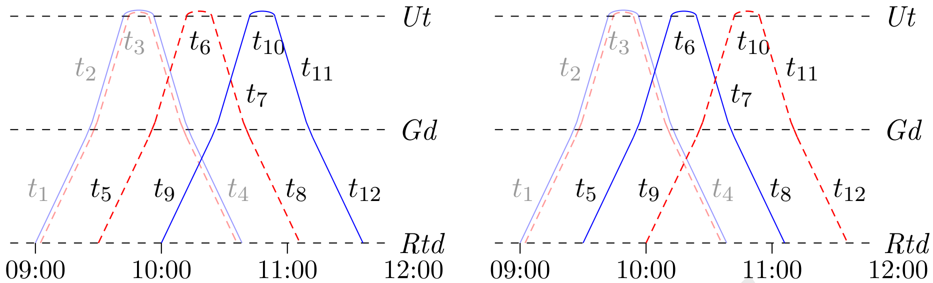


Figure 3: A two-opt move on the assigned duties

The algorithm for finding all two-opt duty exchanges is formalized in Algorithm 1. In this algorithm, $\texttt{coupled}(d, c)$ indicates whether duty $d \in \mathcal{D}$ is coupled to a composition at transition $c \in \mathcal{C}$. In particular, this means that the train unit corresponding to duty d is moved from the shunting yard and coupled to an outgoing trip of transition c. Moreover, $\texttt{findAvailableDuties}()$ gives a map $f : \mathcal{C} \to 2^{\mathcal{D}}$ that lists for each transition $c \in \mathcal{C}$ the duties that are available for coupling. Note that we only need to consider a swap of duties at a transition if at least one of them is coupled at this transition, as we can consider the same swap at some transition later in time if this is not the case.

Algorithm 1: TwoOptDutyNeighborhood

$moves \leftarrow \emptyset$;
$f \leftarrow \texttt{findAvailableDuties}()$;
for each $c \in \mathcal{C}$ **do**
  for each $\{d_1, d_2\} \subseteq f(c)$ **do**
    if $\texttt{coupled}(d_1, c) \vee \texttt{coupled}(d_2, c)$ **then**
      $moves \leftarrow moves \cup \{(d_1, d_2)\}$;
return $moves$;

Note that Algorithm 1 requires at most $\mathcal{O}(|\mathcal{C}||\mathcal{D}|^2)$ steps, as we can find the available duties by looping over all duties in time $\mathcal{O}(\mathcal{D})$. However, the average number of steps needed by Algorithm 1 is often significantly lower, as normally only a handful of duties are present at the same station at any moment in time. Moreover, the computation time can be further reduced by storing in memory the results of $\texttt{findAvailableDuties}()$ and updating the cached results when a new circulation is found.

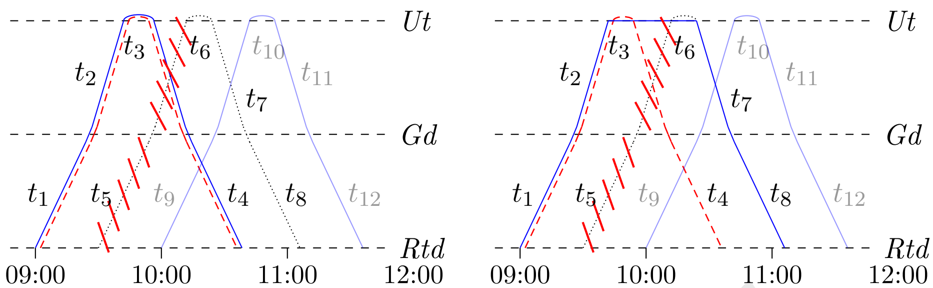


Figure 4: An adjustment of the assigned duties for the blue (solid) rolling stock type. The initial disruption is given by the strikethrough line, while a dotted line indicates that no rolling stock is assigned.

4.2 The Adjusted Path Neighborhood

The main idea behind the Adjusted Path neighborhood is to improve the circulation for one rolling stock type at a time. This implies that we fix the assignment of rolling stock for all-but-one rolling stock types and then improve the assignment for the remaining rolling stock type. We can then exploit the similarity of this simpler problem to a min-cost flow problem to explore this neighborhood efficiently, using an idea inspired by that of Budai et al. (2010).

An example of a move in this neighborhood is shown in Figure 4, where we consider a disruption that leads to the cancellation of trips t_5 and t_6 and to no train units being assigned to operate trips t_7 and t_8. Due to the disruption, the transition between trips t_6 and t_7 is updated. In the shown move, we adjust the duty of one blue (solid) train unit, which now operates trips t_7 and t_8 instead of trips t_3 and t_4. In particular, the train unit is uncoupled after trip t_2, moved to the shunting yard, and then coupled to t_7. Note that the move does not alter the duties for the red (dashed) train units. The result of this rescheduling action is that we can prevent two of the four trip cancellations, but also that the number of seats on trip t_3 and t_4 is reduced.

By fixing the assignment of all rolling stock types except one, say type a, compositions can only be changed by adding or removing train units of type a from the composition. As we leave the assignment for other types unaltered, we can thus represent each composition by only looking at the number of units of type a that are present at the different positions in the composition. For example, the composition abb, consisting of one train unit of type a in the back and two train units of type b in the front, can only be changed by adding or removing train units at the front of the composition, between the two units of type b and at the end of the composition. Each composition that we consider in this neighborhood for this trip is thus of the form

$$a \cdots a\ b\ a \cdots a\ b\ a \cdots a, \tag{1}$$

where we still need to determine the number of units of type a in each group of units of type a. Note that there can also be zero train units of type a in a group, which is, for example, the case for the first two groups in composition abb. This group representation of a composition has been introduced by Budai et al. (2010).

More formally, let g_t be the number of groups of type a in the current composition of trip t. Each composition for trip t can then be represented as a vector $a_t \in \mathbb{Z}_+^{g_t}$, which describes the number of train units of type a present in each group of train units of type a. The problem we consider then becomes to assign feasible vectors a_t to each trip $t \in \mathcal{T}$, such that we use no more train units than available and such that feasible composition changes can be found for each of the transitions. In this way, the assignment of train units for any other type than a remains unaltered.

4.2.1 A Graph Representation

Based on the above group representation, we can consider the duties of the train units of the considered type to represent a flow in a suitable directed acyclic graph $D = (V, A)$. First, we consider the set of nodes V. For each trip $t \in \mathcal{T}$ and each group $g \in \{1, \ldots, g_t\}$ we add nodes $v_{t,g}^{\text{dep}}$ and $v_{t,g}^{\text{arr}}$ representing the departure and arrival of train units in a group of the trip respectively. Moreover, we introduce for each transition $c \in \mathcal{C}$ nodes v_c^{unc} and v_c^{cou}, representing the uncoupling and coupling of train units at this transition respectively.

Second, we consider the set of arcs A. For each trip $t \in \mathcal{T}$ and each group $g \in \{1, \ldots, g_t\}$ we add an arc $(v_{t,g}^{\text{dep}}, v_{t,g}^{\text{arr}})$ to represent that train units are part of this group for the given trip. Furthermore, we add an arc $(v_{t,g}^{\text{arr}}, v_{t',g'}^{\text{dep}})$ for subsequent trips t and t' if there are train units that can operate in group g' of trip t' after operating in group g of trip t. Moreover, we add an arc $(v_{t,g}^{\text{arr}}, v_c^{\text{unc}})$ if train units from group $g \in \{1, \ldots, g_t\}$ can be uncoupled after trip $t \in \mathcal{T}_c^-$ in transition c. Similarly, we add an arc $(v_c^{\text{cou}}, v_{t,g}^{\text{dep}})$ if train units can be coupled to group $g \in \{1, \ldots, g_t\}$ before trip $t \in \mathcal{T}_c^+$ in transition c. Lastly, the coupling and uncoupling nodes are connected by an arc if they occur at the same station and are consecutive in time.

An example of the resulting graph is now given in Figure 5 for the blue (solid) train unit type and the circulation as considered in Figure 2. The compositions assigned to trips t_1 to t_4 and t_5 to t_8 both contain one red (dashed) train unit. Because of that, there are two blue groups for these trips, as depicted in Figure 5. There is only one blue group for trips t_9 to t_{12}, as the composition for these trips consists of a single blue train unit. Moreover, note the arcs that represent the coupling and uncoupling of train units, which connect the nodes at the stations to the trip nodes.

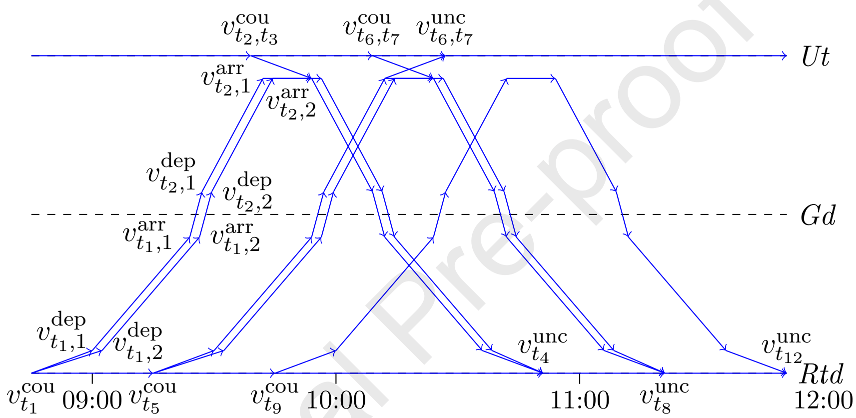


Figure 5: Graph representation for the blue (solid) train unit type for the circulation in Figure 2. For clarity, only a selection of the node labels is included in the figure.

4.2.2 An Augmenting Path Approach

The current circulation can now be represented as a flow in the graph D. However, not every flow in D will correspond to a feasible flow. For a flow to be feasible, the flow should satisfy the restrictions on the compositions that can be used for the trips and on the composition changes that can be used for the transitions. These restrictions can, in general, not be translated to capacity restrictions on single arcs.

A second complicating factor is that the costs of using an arc are in general not linear in the amount of flow over the arc. Note, for example, that adding a train unit of the considered type a gives a far larger benefit when there is currently a shortage of seats than when there are already enough seats. In the latter case, adding a train unit may even lead to a worse objective value.

However, an interesting observation is that we are still able to determine the cost of adding a train unit to a composition and that of removing a train unit from the composition. Let $f(a_t)$ be the cost of using the composition implied by assignment $a_t \in \mathbb{Z}_+^{g_t}$. Then, the change in cost of adding a train unit to group j is given by

$$f(a_t + e_j) - f(a_t)$$

where e_j is a unit vector with a single one at element j. Similarly, the cost of removing a train unit from group j is given by

$$f(a_t - e_j) - f(a_t).$$

In a similar way, we can determine the change to the costs for changing the number of coupled and uncoupled units at a transition.

Moreover, a second observation is that when adding a train unit or removing a train unit from the composition, we can determine if the new composition is feasible. In essence, we can determine if the compositions given by $a_t + e_j$ and $a_t - e_j$ are feasible. Similarly, we can determine if the new composition change that is is formed by adjusting the flow on one arc in the composition change remains feasible.

The above two observations motivate to look at the residual graph $D' = (V, A')$. In particular, for each arc $(u, v) \in A$, we add the arc (u, v) to A' if we can increase the flow on this arc by a single unit. Furthermore, we add the arc (v, u) to A' if it is possible to decrease the flow on arc (u, v) by a single unit. Let these backward and forward arcs respectively be given by the sets A^* and A^{**}. Moreover, we can assign to each arc a cost that corresponds to the new composition or composition change that is formed.

We argue now that finding an improvement in the assignment of the corresponding rolling stock type corresponds to finding a cycle with negative cost in graph D', where the arc costs are path-dependent. In particular, if a path uses more than one arc corresponding to the same trip or transition, the total arc costs are usually not equal to the sum of the arc costs. For example, it is generally the case that

$$f(a_t + e_i - e_j) \neq f(a_t + e_i) + f(a_t - e_j).$$

Hence, we have to take into account any arcs that are already in the path to determine the cost of a newly encountered arc.

4.2.3 Finding Negative Weight Cycles

As the problem of finding a negative weight cycle with path-dependent arc costs is $\mathcal{NP}$-hard, we use a heuristic approach to find such cycles here. To find negative weight cycles we split the problem into that of freeing an existing duty and finding a new duty for the considered train unit. In particular, this corresponds to first finding a path in D' that only uses backward arcs. For the ending point of this path, a new duty is then found by finding a path from the ending point of this backward path to the starting point of this backward path that only uses forward arcs. This thus restricts the considered cycles to those that can be formed by one backward path and one forward path.

Let $P \subseteq V$ now represent the set of possible starting points of a forward path of a train unit. Such a starting point corresponds to a coupling node v_c^{cou} if a train unit is parked at a station at the start of the planning horizon. Alternatively, it corresponds to a trip node $v_{t,g}^{\text{dep}}$ if a train unit is operating in this group of this trip at the start of the planning horizon. Moreover, let $D^* = (V, A^*)$ and $D^{**} = (V, A^{**})$ be the graphs containing respectively the backward and forward arcs. In addition, let s' be a source and sink node for the backward and forward graph respectively that is connected to the last station node of each station.

The procedure that is used to explore the neighborhood is now formalized in Algorithm 2. In this algorithm, we first fix the train unit type for which we try to find an improvement in the assignment. We then generate for each possible starting node a backward path from the sink node of the graph to the starting point. If such a backward path can be formed, we find a matching forward path from the starting node to the sink node. Together, this backward path and forward form a cycle in the original graph D'. Note that the backward path can be interpreted here as freeing up part of an existing duty for rescheduling, while the forward path can then be seen as finding a new completion of the duty for this train unit for the remainder of the planning horizon.

Note that Algorithm 2 requires at most $\mathcal{O}(|\mathcal{R}||\mathcal{D}||\mathcal{T}|)$ steps. In particular, we execute the algorithm for each rolling stock type $r \in \mathcal{R}$. Moreover, the number of starting points is no more than the number of duties $|\mathcal{D}|$, as each duty is contained in at most one starting point and as we only need to consider those starting points where at least one train unit starts. Lastly, as we consider directed acyclic graphs when finding shortest paths, each shortest path algorithm takes at most $\mathcal{O}(|\mathcal{T}|)$ steps if we assume some fixed maximum length for the compositions. In practice, we can generally speed

Algorithm 2: AdjustedPathNeighborhood

$moves \leftarrow \emptyset$;
for each $r \in \mathcal{R}$ **do**
  $D^* \leftarrow$ `createBackwardGraph`(r);
  for each v **do**
    $P \leftarrow$ `findShortestPath`(D^*, s', v);
    if $P \neq \emptyset$ **then**
      $D^{**} \leftarrow$ `createForwardGraph`(r, P);
      $P' \leftarrow$ `findShortestPath`(D^{**}, v, s');
      if $P' \neq \emptyset$ **then**
        $moves \leftarrow moves \cup \{(P, P')\}$;
return moves;

up the computations. For example, as the backward graph D^* is the same for all backward paths, all single-source shortest paths can be derived at the same time with $\mathcal{O}(|\mathcal{T}|)$ steps. Moreover, as the considered graph is different for each train unit type, we can parallelize the algorithm over the train unit types.

4.3 Composition Change Neighborhood

The Composition Change neighborhood adjusts the composition changes at transitions directly. The main motivation for using this neighborhood is that the other neighborhoods tend to only change a few duties at a time. However, as costs are associated with the used composition changes and especially the shunting actions performed in them, these neighborhoods may fail at changing the composition changes in such a way that high costs are avoided.

An example of how a change to a composition change looks is given in Figure 6. In this example, originally two units are uncoupled at the transition between trip t_6 and t_7, which are later used to operate trips t_{11} and t_{12}. The composition change for this transition is then changed into one where no shunting occurs and where all train units continue to operate on trips t_7 and t_8. This change might, for example, be beneficial when the composition change before the disruption was also one where no shunting was performed.

If a composition change is altered, the circulation has to be changed ac-

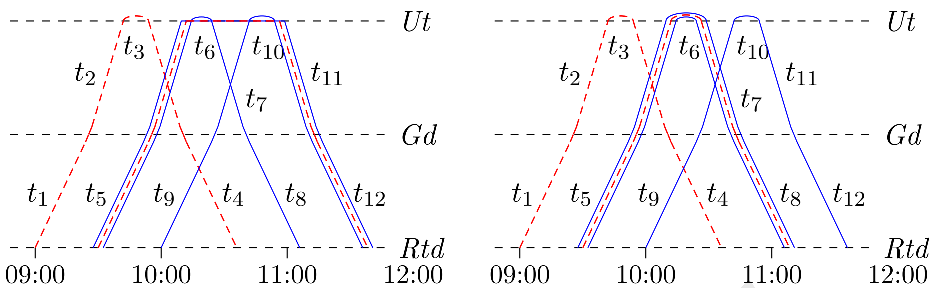


Figure 6: Example of changing the composition change for a transition.

cordingly. To do so, we employ an idea similar to the one in the Adjusted Path Neighborhood as introduced in the previous section. In particular, we find new duties that match the adjustments that are made to the composition change. These duties then define a new circulation.

Let the current composition change for some transition $c \in \mathcal{C}$ be given by $q \in \mathcal{Q}_c$. Moreover, consider that we want to change q into some new composition change $q' \in \mathcal{Q}_c$. We will require in this neighborhood that all compositions that occur before this transition remain unaltered. This implies that a $q' \in \mathcal{Q}_c$ needs to be found of which the composition $q'(t)$ on the incoming trip $t \in \mathcal{T}_c^-$ is unaltered compared to the current circulation.

The above requirement implies that q' varies from q in the shunting that takes place. We can then identify which train units on the incoming and outgoing trip are affected by this change in shunting. For incoming trips, those are the train units for which the uncoupling is changed, while for outgoing trips those are the train units for which the coupling is changed. We can then create new duty templates for each of the affected train units, which describe the action that is taken in this transition.

To complete these duty templates into actual duties, we use again the group representation graph as introduced in the Adjusted Path Neighborhood. In particular, we find for each duty template a starting node in this graph. We then find a backward path from the sink node to this starting node that frees up an existing duty. If such a path can be found, we find a forward path that completes this freed up duty.

The complete algorithm is given in Algorithm 3. In this algorithm, the function `FindAdjustedDuties`(c, q) determines the duty templates that are needed for the new composition change. Furthermore, the function `findRelevantNode`(c, d) finds the correct starting node for this duty in the graph representation of this transition. Note that we can only find a move in this algorithm if we find feasible backward and forward paths for each of these duty templates that need to be completed.

Algorithm 3: Composition Change Neighborhood

```
moves ← ∅;
for each c ∈ 𝒞 do
    for each q′ ∈ 𝒬_c do
        currentMove ← ∅;
        pathsFound ← true;
        𝒟′ = FindAdjustedDuties(c, q′);
        for each d ∈ 𝒟′ do
            D* ← createBackwardGraph(d);
            v ← findRelevantNode(c, d);
            P ← findShortestPath(D*, s′, v);
            if P ≠ ∅ then
                D** ← createForwardGraph(d);
                P′ ← findShortestPath(D**, v, s′);
                currentMove ← currentMove ∪ {(P, P′)};
            else
                pathsFound ← false;
        if pathsFound then
            moves ← moves ∪ {currentMove};
return moves;
```

Note that Algorithm 3 requires at most $\mathcal{O}(|\mathcal{C}||\mathcal{Q}||\mathcal{D}||\mathcal{T}|)$ steps, which follows from each shortest path being found with complexity $O(|\mathcal{T}|)$ due to the graphs being directed acyclic graphs. However, the number of needed steps is generally considerably less, as for only a few duties the shunting is altered at a transition. Moreover, only a few composition changes have the required predecessor compositions. As a result, this neighborhood can be explored in a reasonable amount of time even for larger problem instances.

4.4 The VNS heuristic

We combine the above neighborhoods into a Variable Neighborhood Search (VNS) heuristic (Mladenović and Hansen 1997). In VNS, we first perturb the current circulation in each iteration, to then apply a local search procedure to improve the perturbed circulation. Perturbation occurs through picking a new circulation randomly from one of the shaking neighborhoods, where we switch to the next neighborhood if no improvement is found in the current one. The general outline of our VNS heuristic is given in Algorithm 4.

Algorithm 4: The VNS heuristic

```
Input: A starting solution c, neighbourhoods N_1, ..., N_{k_max}
while ¬StopConditionSatisfied() do
    k ← 1;
    while k < k_max do
        c' ← Shake(c, N_k);
        c'' ← VariableNeighborhoodDescent(c');
        if f(c'') < f(c) then
            c ← c'';
            k ← 1;
        else
            k ← k + 1;
```

For local search within our VNS heuristic, we employ Variable Neighborhood Descent. In Variable Neighborhood Descent, we explore multiple neighborhoods in a deterministic way. There is thus no diversification and the local search stops when none of the neighborhoods can provide an improvement to the current solution. In our heuristic, we only use the Two-Opt Duty Neighborhood N_{duty} and the Adjusted Path Neighborhood N_{path} in Variable Neighborhood Descent, where we use the order (N_{duty}, N_{path}) to explore these two neighborhoods. Moreover, we employ a best improvement strategy, where we always explore the complete neighborhood to find the move which results in the largest improvement. Note that computing the improvement made by a move, and checking its feasibility, can be done in $\mathcal{O}(\mathcal{T})$ steps.

As shaking neighborhoods, we then use the Adjusted Path and Composition Change neighborhood in the order (N_{path}, N_{change}), where N_{change} is the Composition Change Neighborhood. For both shaking neighborhoods, we draw a move at random from all possible moves that can be made in the neighborhood. By focusing on these two larger sized neighborhoods, we try to ensure that enough variation is present in the found solutions to ensure that local optima are escaped. Moreover, by only sampling from the Composition Change Neighborhood, we prevent the relatively large overhead of finding all moves within this neighborhood.

As a stopping criterion, we solely use a maximum elapsed time. This reflects the idea that the heuristic could be stopped at any moment in time by a rolling stock dispatcher to obtain a solution. Hence, if very little time is available, a rolling stock dispatcher can stop the search process and evaluate the solution that is available at that moment in time. Note that this solution is always feasible, as we only consider moves that leave the feasibility of the circulation intact.

5 Rolling Stock Rescheduling With Flexible Turning

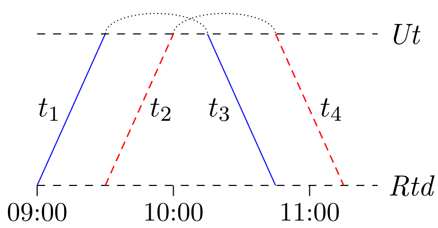
In this section, we extend the heuristic to a setting where the transitions between trips at the ending stations are no longer fixed. Instead, these transitions at ending stations can be changed, implying that incoming trips can be freely reassigned to outgoing trips. This setting for rolling stock rescheduling is often referred to as flexible turning (Nielsen 2011).

5.1 Problem Definition

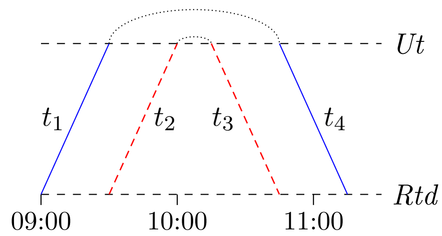
The transitions $\mathcal{C}$ can generally be split into those transitions which occur at in-between stations of a railway line and those that occur at terminal stations. We will refer to the latter as turnings, as trains are often turned around during these transitions (see Table 1). Let $\mathcal{C}' \subseteq \mathcal{C}$ give the set of turnings. Unlike transitions at in-between stations, where some passengers often remain in the train to continue their journey towards the terminal station, no passengers remain within the train during turnings. Hence, it is generally possible to change these turnings for a rolling stock dispatcher when faced with a disruption.

Changing these turnings at ending station essentially corresponds to re-assigning incoming trips to other outgoing trips. An example is shown in Figure 7. Note how there is originally a turning present between trips t_1 and t_3, and between trips t_2 and t_4. After rescheduling, this is changed to a turning between trips t_1 and t_4, and between trips t_2 and t_3. The impact of this rescheduling action is that the compositions on the outgoing trips are reversed, which impacts the further circulation as well when these train units move on to successor trips later in the planning horizon.

We consider a similar setting as Nielsen (2011) for flexible turning. Here, compositions that arrive on incoming trips can be allocated to any outgoing trip that leaves from the same stations after some fixed minimal turning



(a) Original turnings



(b) Alternative turnings

Figure 7: Example of changing the turnings at a terminal station

time. Moreover, for each incoming and outgoing trip of a turning, some fixed set of shunting actions can take place after and before the trip, respectively. The composition that arrives from an incoming trip, after shunting has taken place, then remains at the platform of a station until it leaves again, possibly again after shunting takes place, on an outgoing trip.

The rolling stock rescheduling problem with flexible turning can then be formulated as the problem of assigning compositions to the trips, composition changes to the transitions and turnings for the ending stations. Next to having to satisfy all requirements of the rolling stock rescheduling problem, we now additionally need to ensure that each incoming trip in a turning is assigned to a feasible outgoing trip. The objective of this problem is similar to that of the rolling stock rescheduling problem, with the difference that we generally want to penalize any changes that are made to the turnings in order to minimize the impact on the station plans and the crew duties.

5.2 k-Opt Turning Neighborhood

To incorporate flexible turning into our VNS heuristic, we add a neighborhood. This neighborhood consists of a k-opt procedure for the turnings at the stations, in which we reassign the turnings between k incoming and outgoing trips of a station. Moreover, a repair step is used to adapt the circulation to the changes that are made to the turnings.

Consider the set of turnings $\mathcal{C}'_s \subseteq \mathcal{C}'$ at station $s \in \mathcal{S}$. For a k-opt move, with $k \in \mathbb{Z}_+$ we then consider a set $\mathcal{C}^* \subseteq \mathcal{C}'_s$ with $|\mathcal{C}^*| = k$. Let $\mathcal{T}^-_{\mathcal{C}^*} = \{t \mid t \in \mathcal{T}^-_c, c \in \mathcal{C}^*\}$ and $\mathcal{T}^+_{\mathcal{C}^*} = \{t \mid t \in \mathcal{T}^+_c, c \in \mathcal{C}^*\}$ denote the set of incoming trips and outgoing trips for the turnings in $\mathcal{C}^*$, respectively. Each possible way of reassigning the turnings in $\mathcal{C}^*$ is then given by a perfect matching $g : \mathcal{T}^-_{\mathcal{C}^*} \to \mathcal{T}^+_{\mathcal{C}^*}$, where an incoming trip can only be matched to an outgoing trip when enough time is available to execute the turning.

As an example, a 2-opt move on the turnings in Figure 7 has two possible matchings, which are the two cases represented in the figure. Note that one of the possible turning matchings is always the original matching and that the total number of matchings is dependent on the departure and arrival times of the incoming and outgoing trips. Moreover, note that, as in Figure 7, each matching can lead to different compositions on the outgoing trips. As these do generally not fit with the remainder of the circulation, we need to repair the circulation based on these new turnings.

In the repair step, we essentially propagate the new compositions on the outgoing trips until the end of the planning horizon. More specifically, we determine for each composition on the outgoing trips the remaining trips it executes during the planning horizon by following the transitions in the timetable. We then assign this composition to all of these trips and pick according composition changes. Moreover, we adjust the compositions of any trips that would originally receive rolling stock from the train units on trips in $\mathcal{T}^{+}_{\mathcal{C}^*}$. Note that the repair step can fail when we are unable to pick a feasible composition for a trip or a feasible composition change for a transition.

The complete algorithm is shown in Algorithm 5. In this algorithm, the function `PossibleAssignments`$(\mathcal{C}^*)$ determines all possible matchings for some set of turnings $\mathcal{C}^*$. Moreover, the function `Repair`$(\mathcal{C}^*, g)$ tries to repair the circulation for some set of turnings $\mathcal{C}^*$ and matching g. Note that if the repair step fails, we disregard the current matching.

Algorithm 5: k-Opt Turning Neighborhood

```
moves ← ∅;
for each s ∈ S do
    for each C* ⊆ C'_s : |C*| = k do
        G ← PossibleAssignments(C*);
        for each g ∈ G do
            s ← Repair(C*, g);
            if s ≠ ∅ then
                moves ← moves ∪ {(C*, s)};
return moves;
```

Algorithm 5 takes at most $\mathcal{O}(|\mathcal{S}||\mathcal{C}'|^k|\mathcal{T}|)$ steps. To see this, note that there are at most $\mathcal{O}(|\mathcal{C}'|^k)$ options of picking a set $\mathcal{C}^*$ for each station and that the number of possible options to reassign the turnings is only dependent on k and is therefore constant. Moreover, note that in the repair step we iterate forward over the trips, thus requiring no more than $\mathcal{O}(|\mathcal{T}|)$ steps.

5.3 VNS Heuristic

We consider a similar set-up as before for our VNS heuristic, but adjust the neighborhoods that we use. In particular, we add the 2-opt turning neighborhood both to the existing local search neighborhoods and to the existing shaking neighborhoods. Initial experiments have shown that picking $k = 2$ in both cases leads to a good trade-off between solving time and size of the neighborhood. In both cases, the 2-opt turning neighborhood is the last neighborhood to be explored.

6 Computational Results

In this section, we test the heuristic on instances of Netherlands Railways (NS). Our aim is to evaluate the quality of the circulations that are provided by the heuristic, which we do by comparing them to the circulations obtained by the exact solution method of Fioole et al. (2006). Next to looking at the quality of the circulations, we also investigate the performance of the considered neighborhoods. Moreover, we test how the extension of the heuristic to flexible turning performs by comparing the heuristic to the exact method that was proposed by Nielsen (2011) for this setting. Before we look at these numerical results, we introduce the considered rolling stock rescheduling instances and the objective function that is used to evaluate the found rolling stock circulations.

6.1 Instances

Our instances are derived from the timetable that was operated by NS in 2018. NS is the largest passenger railway operator in the Netherlands and operates both Intercity and regional (Sprinter) services throughout the country. The network that was operated in the Netherlands by NS in 2018 is shown in Figure 8. As the rolling stock planning of NS considers a planning horizon of a day, we consider the timetable as it was on a Tuesday, as Tuesday has on average the highest passenger numbers and is thus seen as the hardest day of the week to plan for.

Instance classes of varying size are created by selecting subsets of the rolling stock types and by including those trips that can be operated by the selected types. An overview of the instance classes is given in Table 2,

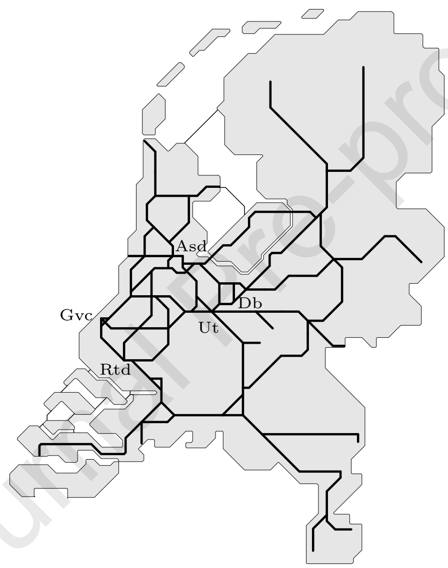


Figure 8: The railway network operated by NS in 2018. Included are the four largest stations: Utrecht Centraal (*Ut*), Amsterdam Centraal (*Asd*), Rotterdam Centraal (*Rtd*) and Den Haag Centraal (*Gvc*). Moreover, station Driebergen-Zeist (*Db*) is depicted.

where we describe the included train unit types and give summary statistics about the size of these instance classes. Note that the size of the considered instance classes varies significantly when we add additional rolling stock types.

Table 2: Overview of the different instance classes and their properties. Reported are the names of the considered rolling stock types, number of train unit types, number of trips, number of transitions, number of available train units, the average number of allowed compositions per trip and the average number of allowed composition changes per transition.

	Rolling Stock Types	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{D}\rvert$	$\bar{\mathcal{P}}_t$	$\bar{\mathcal{Q}}_c$
Intercity	ICM	2	1549	1686	124	14.01	23.26
	ICM-VIRM	4	3764	4031	271	13.15	18.74
	ICM-VIRM-DDZ	6	4241	4560	306	15.23	22.55
Sprinter	SLT	2	1831	1932	113	7.28	8.30
	SLT-SGM	4	2997	3154	182	10.29	11.28
	SLT-SGM-FLIRT	6	3782	3970	229	11.51	12.27

For each of the above instance classes, we create a rolling stock circulation with the exact mixed integer programming (MIP) model as proposed by Fioole et al. (2006), which is also at the heart of rolling stock scheduling at NS. Using an exact approach to determine the circulation for the undisturbed situation allows us to find a circulation according to the same objectives as in rescheduling. This prevents that improvements can be made to the original circulation even though no disruptions are faced.

6.1.1 Considered Disruptions

The actual rescheduling instances for a given instance class are now created by incorporating the effect of a disruption on the timetable. We consider two types of disruptions that lead to trip cancellations: small train disruptions that only cause a few cancellations and larger infrastructure failures that lead to many trip cancellations. In this way, we look at the performance of the heuristic for the different use cases in which the heuristic may be employed by dispatchers.

Small cancellations may, e.g., be the result of small technical failures on a train unit or of missing crew to operate a trip. We generate small cancellation instances by picking from all trips in the timetable one trip uniformly at random. We then cancel this trip and all the trips that follow it until the train would arrive at the terminal station for the current passenger service. In particular, note that the train units that operate the canceled trips are unable to get to this terminal station, which impacts any further trips on which these train units were planned to be operated as well.

Secondly, we consider larger disruptions by looking at infrastructure failures in which all tracks between two stations become blocked. Such blockages occur, e.g., when the overhead power lines become damaged on a section of railway infrastructure or when there is a defect in the signaling system for that section of infrastructure. To ease the work for dispatchers, the Dutch infrastructure manager has available a contingency plan for such section blockages that specifies the adjustments that need to be made in the timetable. We use these contingency plans to create a new timetable for a rescheduling instance.

A starting circulation for our VNS heuristic is created for each rescheduling instance by employing some simple rules to incorporate the effect of the disruption. In essence, we propagate the existing compositions for those trips that have been affected by the disruption, i.e., which are operated by rolling stock of which the duty is affected by the disruption. Moreover, if a trip would originally pick up rolling stock units from the shunting yard and if the duties of those train units are disturbed, then we cancel those trips in the starting circulation. Note that while the cancellations that follow from the disruption can no longer be prevented, it is the aim in the heuristic to prevent these knock-on cancellations.

6.1.2 The Rescheduling Setting

As we consider a real-time rescheduling setting, the information about a disruption only comes in during the day of operation. This implies that a part of the circulation has already been executed and that no changes can be made anymore to that part of the circulation. Moreover, some time is needed to make the decisions and to communicate any changes that are made. As a result, we fix the duties of the rolling stock units up until 30 minutes after the disruption starts. The goal is then to reschedule the circulation for the remainder of the day.

6.2 Objective Function

An overview of the considered cost parameters in the objective function is given in Table 3. The first three cost components consider the costs that follow as a result of the chosen compositions. First, a large penalty is incurred in the objective function if a trip is canceled, which corresponds to assigning an empty composition to a trip. Second, a penalty is incurred for each passenger that is expected to have to stand on a trip. This penalty is incurred per standing passenger and per km of distance on the trip. Lastly, a penalty is incurred for the use of the train units, which is incurred per kilometer that a train unit is used and is scaled with the length of the train unit in carriages.

Table 3: Cost parameters in the objective function.

Element	Objective	Weight
Composition	Cancellation	1000000
	Seat Shortage	0.2
	Mileage	0.1
Shunting	New Shunting	1000
	Changed Shunting	100
	Canceled Shunting	50
Inventories	Ending Inventory Deviation	10000
Transitions	Flexible Turning	1000

The next elements in the objective function relate to the costs that follow from the shunting that occurs at composition changes. Changing the shunting pattern at stations, i.e., at which transitions uncoupling and coupling occurs, may be costly, as it implies that also the local plans at the stations need to be altered. It may, e.g., require additional crew to execute the shunting and additional parking space at the shunting yard to store any uncoupled train units. In the objective function, we differentiate between a new shunting action for a composition change, changing the shunting at a composition change and canceling the shunting at a composition change.

Moreover, any deviations in the ending inventories are penalized, which implies that a penalty is incurred when the number of train units that end at a station deviates from the target number of train units to end there at the end of the planning horizon. This penalty is incurred per unit of deviation for each train unit type and each station. Note that this penalty resembles the costs that a railway passenger operator has to incur to rebalance any deviations overnight.

The last cost parameter relates to the use of flexible turning, which we will consider in Section 6.6. Here, we penalize a turning if it is not contained in the original rolling stock circulation. This is done to prevent that too many turnings are changed, which can cause problems in the crew rescheduling phase as crews often stay on the train during turnings. The cost is incurred per changed turning.

6.3 Results for Small Disruptions

In this section, we look at the performance of the heuristic on the small disruption instances with fixed rolling stock turnings. To evaluate the performance of the heuristic, we compare the circulations found by the heuristic after one minute of solving time to the optimal circulations obtained by the exact method of Fioole et al. (2006). The experiments have been run on a computer with an Intel Xeon Gold 6130@2.1Ghz processor and 96GB of internal memory. Moreover, CPLEX 12.9 was used to solve the model of Fioole et al. (2006) and the heuristic was programmed in the Java programming language. The obtained results are given in Table 4, where 50 instances, each corresponding to a different trip being canceled, have been run for each instance class. Moreover, Figure 9 shows the progress of the heuristic over the given computation time.

Table 4: Results for the small disruptions. Shown for both methods are the obtained objective (*Object.*) and the number of canceled trips (*Canc.*). Moreover, we state the solving time needed by the exact method (*Time*). All these results are averaged over 50 instances. Moreover, we report for the heuristic the absolute objective gap compared to the exact method, where we state the minimum, median, average and maximum gap.

	Exact Method			Heuristic					
						Absolute Gaps			
	Time (s)	Object.	Canc.	Object.	Canc.	min	med.	ave.	max
ICM	1.74	2156767	2.08	2300299	2.22	0	44	143531	3042469
ICM-VIRM	3.46	1420366	1.24	1586121	1.40	0	1060	165755	2027072
ICM-VIRM-DDZ	5.87	1945470	1.74	1991549	1.78	0	222	46079	2027482
SLT	0.45	3630339	3.56	3852331	3.78	0	0	221992	4013357
SLT-SGM	1.16	4166925	4.08	4810782	4.72	0	3	643857	14016779
SLT-SGM-FLIRT	1.54	4266672	4.16	4289680	4.18	0	0	23007	997668

The results in Table 4 show that the performance of the heuristic varies over the instance classes. For most of the instances, the quality of the circulations found by the heuristic is very close to that of the optimal solutions found by the exact method. This is illustrated by the rather low median

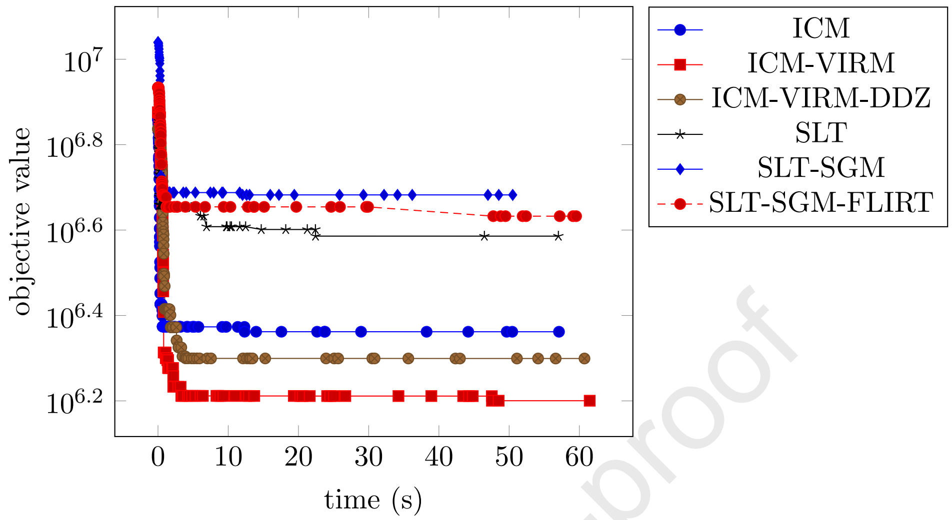


Figure 9: Evolution of the objective of the best found solution by the heuristic over time.

objective gap as obtained for the instance classes, especially in relation to the total objective value. In particular, we see that for the SLT and SLT-SGM-FLIRT instance classes we even find the optimal solution for most of the instances within those classes. Moreover, the number of cancellations, which is by far the most dominant factor in our objective function, is for many of the instance classes rather similar for both methods.

At the same time, we see that there are some instances for which the heuristic is unable to match the performance of the exact method. This is illustrated by the rather high maximum gaps that are obtained for the instance classes. This effect can be contributed to the high penalty assigned to canceling a trip, implying that being unable to prevent a single cancellation, which can be prevented in the exact method, leads to an immediate increase of 1000000 in the obtained objective value. Note that such a difference in the number of cancellations is to be expected for some instances, as the number of changes that need to be made in the circulation to arrive at the minimum number of cancellations might be large, making it significantly more difficult to find such a solution for a heuristic method.

When looking at the performance of the heuristic over the allotted time, as shown in Figure 9, we see that by far the largest improvements are made in the first few seconds of the search process. This can be explained by the large number of cancellations that are present at the start of the solving procedure, as numerous trips may no longer have a composition assigned as a result of the initial cancellation. As some of these cancellations can easily be resolved, much progress is made in the first steps of the heuristic. On the other hand, some of the remaining penalty might be hard to reduce, as shown in the tails of the graph for each of the instance classes.

6.4 Results for Large Disruptions

In this section, we look at the results for the large disruptions that correspond to section blockages. We again compare the circulations obtained by the heuristic to those obtained by the exact method of Fioole et al. (2006) and use a similar computational setup as in the previous section. In particular, note that we again use a stopping criterion that corresponds to one minute of solving time for the heuristic. The results for the large disruptions are given in Table 5, where we consider a section blockage between the stations Utrecht Centraal (*Ut*) and Driebergen-Zeist (*Db*) for the timetable as considered in the ICM-VIRM-DDZ instance class. Note that, opposed to the last section, only one instance is considered for each entry, where each instance corresponds to a different time-frame during which the relevant infrastructure section is blocked.

Table 5: Results for the large disruptions. For each instance we state the percentage of fixed trips due to the moment at which the disruption takes place. For both methods we state the obtained objective (*Object.*) and the number of canceled trips (*Canc.*). For the exact method, we also give the solving time (*Time*). For the heuristic, we also state the objective gap compared to the exact method (*Gap*).

		Exact Method			Heuristic		
Instance	Fixed (%)	Time (s)	Object.	Canc.	Object.	Canc.	Gap
Ut-Db: 07-08	11	18.54	6311061	6	8405433	8	2094372
Ut-Db: 09-12	22	12.72	7262961	7	7351712	7	88751
Ut-Db: 12-14	38	7.49	8268163	8	8349301	8	81138
Ut-Db: 16-18	60	4.34	8321181	8	8369584	8	48403
Ut-Db: 21-22	87	2.81	4251822	4	4251841	4	18

The results in Table 5 are mostly in accordance with those found in Table 4. We see that for all but one instances the number of cancellations for the heuristic is equal to the number of cancellations for the exact method. As the prevention of cancellations is the most important objective in our problem, this implies that also the objective value is relatively similar between the heuristic and exact method for all these instances. The exception is the 07-08 instance, for which there are 2 additional cancellations in the solution as obtained by the heuristic and for which there is thus a larger difference in the objective value between the heuristic and the exact method.

An interesting observation from the results in Table 5 is that the quality of the circulations found by the heuristic tends to improve when the disruption occurs at a later moment in time. This can be explained by the impact that the starting time has on the size of the instance, as the circulation is considered fixed until the start of the disruption. In particular, we see that 87% of the trips can no longer be changed when the disruption occurs from 21:00 - 22:00. Hence, these results show that the heuristic tends to perform better for instances which have more limited search space.

6.5 Performance of the Neighborhoods

To get a better insight into the performance of the heuristic, we look in this section at the contribution of the different neighborhoods to the overall results. To do so, we report summary statistics for both the neighborhoods that we use in local search and for those that we use for shaking. The results that we report correspond to those experiments that were run for the small disruptions. Summary statistics for the local search neighborhoods are given in Table 6 and for the shaking neighborhoods in Table 7.

Table 6: Results for the local search neighborhoods. Reported for each neighborhood are the number of times the neighborhood is explored (*Iter.*), the number of times exploring the neighborhood has led to an improvement (*Impr.*) and the total time spent on exploring this neighborhood (*Time*).

	Two-Opt Duty			Adjusted Path		
	Iter.	Impr.	Time (s)	Iter.	Impr.	Time (s)
ICM	581	15	4	566	177	40
ICM-VIRM	165	6	5	159	62	40
ICM-VIRM-DDZ	161	6	5	154	57	41
SLT	566	28	4	538	206	40
SLT-SGM	252	18	5	234	93	38
SLT-SGM-FLIRT	210	8	5	201	76	39

When looking at the above results, we clearly see the size of the different neighborhoods reflected in the execution time that is spent on these neigh-

Table 7: Results for the shaking neighborhoods. Reported for each neighborhood are the number of times the neighborhood is explored (Iter.) and the total time spent on exploring this neighborhood.

	Adjusted Path		Comp. Change	
	Iter.	Time (s)	Iter.	Time (s)
ICM	197	12	111	3
ICM-VIRM	49	12	41	3
ICM-VIRM-DDZ	49	12	30	3
SLT	166	12	165	4
SLT-SGM	71	12	68	4
SLT-SGM-FLIRT	63	12	62	4

borhoods. In particular, we see that while the Two-Opt Duty Neighborhood is executed more often than the Adjusted Path Neighborhood, far more time is spent on the latter. At the same time, we also observe that the Adjusted Path Neighborhood is far more often able to find improvements, justifying the larger amount of time spent here. Interestingly, the time spent on the Composition Change Neighborhood is limited, which can be explained by the fact that we can efficiently find random moves by iteratively going over the transitions and possible composition changes until we find a potential move.

Another interesting result is that the time spent on the different neighborhoods remains relatively constant over the different instance classes. In particular, we see that we always spent around 40 seconds in local search on the Adjusted Path Neighborhood, while we spent around 4 to 5 seconds on the Two-Opt Duty Neighborhood. This result can most likely be explained by the fact that both neighborhoods depend in a linear way on the number of trips in the timetable and that the number of trips is the most dominant factor in their running time.

6.6 Results for Flexible Turning

To show the performance of the heuristic on rich rolling stock settings, we consider in this section the rolling stock rescheduling problem where flexible turning is allowed at all terminal stations. This implies that we use the version of the heuristic as proposed in Section 5, where we use a fourth neighborhood to incorporate flexible turning. We consider again the setting with small disruptions, as in Section 6.3. Moreover, the benchmark model we use is now the model for flexible turning as proposed by Nielsen (2011), which extends the model of Fioole et al. (2006). The results of our experiments are given in Table 8, where a time limit of 1 minute of solving time is considered for both the heuristic and the exact method. As we found that the exact method is sometimes unable to find a solution within this solving time, and in order to more fairly compare it to the heuristic, we use the same starting solution as in the heuristic to warm start the exact method.

Table 8: Results with flexible turning. For both methods we state the obtained objective (*Object.*) and the number of canceled trips (*Canc.*). Moreover, we state for the exact method the solving time (*Time*) and the number of instances solved to optimality (*Solved*). All results are averaged over 50 instances.

	Exact Method				Heuristic	
	Time (s)	Solved	Object.	Canc.	Object.	Canc.
ICM	6.05	50	2094913	2.02	2199591	2.12
ICM-VIRM	18.41	48	1379904	1.20	1546817	1.36
ICM-VIRM-DDZ	24.14	42	3397814	3.18	1971968	1.76
SLT	6.03	50	1783561	1.72	3150383	3.08
SLT-SGM	20.32	48	1961036	1.88	2828711	2.74
SLT-SGM-FLIRT	18.44	49	3144072	3.04	3809493	3.70

The results in Table 8 show that the exact method is unable to solve all instances to optimality within one minute of solving time when including flexible turning. Especially the instances in the ICM-VIRM-DDZ class turn out to be hard to solve, where the exact method is unable to find an optimal solution for 8 out of the 50 instances. This is also reflected in the average objective value and number of cancellations, which turn out to be significantly higher than for the ICM and ICM-VIRM instance classes. However, we also find that instances from instance classes with fewer rolling stock types can be solved relatively quickly, which is the case for the ICM and SLT instance classes.

When we compare the heuristic to the exact method, we see that the heuristic outperforms the exact method for the ICM-VIRM-DDZ instance class. In particular, we see both a lower average objective value and a lower average number of cancellations for the heuristic on this instance class. For the other classes, we again see a similar image as in the setting without flexible turning. For some instances, in this case especially those of the ICM and ICM-VIRM instance class, we see that the solutions found by the heuristic are close to those found by the exact method. For other instances, in this case especially those of the SLT and SLT-SGM instance classes, we see a somewhat larger gap between the heuristic and the exact method.

To further explore the relative performance of the heuristic and the exact solution method on the ICM-VIRM-DDZ instance class, we show in Figure 10 for each instance the best found objective value by the exact method over the given solving time. Moreover, we show in Figure 11 how the start time of the disruption impacts the solving time of the exact solution method by plotting the start time of the disruption against the solving time for each instance. As we saw that the exact method needs more than one minute of solving time for eight instances in the ICM-VIRM-DDZ instance class, we consider for these figures a longer maximum running time of 30 minutes. Note that such a running time would not be representative for a rolling stock rescheduling setting, but does allow us to analyze if the exact method would perform better with more solving time.

The results in Figure 10 show that there are a significant number of instances for which the exact method is only able to find good solutions after a few minutes of solving time. This is illustrated by the jumps in objective value that can be seen around 3 to 5 minutes of solving time. Moreover, it can be seen that for another two instances optimality cannot be proven within 30 minutes of solving time and that for these same instances significant improvements in objective value are still achieved between 5 and 10 minutes of solving time. Overall, the average objective value for the exact method reduces to 1464904, with on average 1.26 cancellations per instance, when given 30 minutes of solving time. Hence, we see that there are on average about 0.5 cancellations more in the solutions of the heuristic, after one minute of solving time, than could be achieved by the exact method when given a longer solving time.

The results for the exact method over 30 minutes of solving time show that there are instances for which the exact method struggles to find solutions within short running times. When looking at Figure 11, these hard instances are especially ones that correspond to a cancellation early during the morning. This can be explained by the fact that these are also the instances that contain the largest number of trips. Interestingly, especially instances that occur around the morning peak hours lead to the exact solution method hitting the maximum solving time of 30 minutes. This can likely be explained by the fact that additional trips are operated during the peak hours, increasing the possibilities for flexible turning around the time of the disruption.

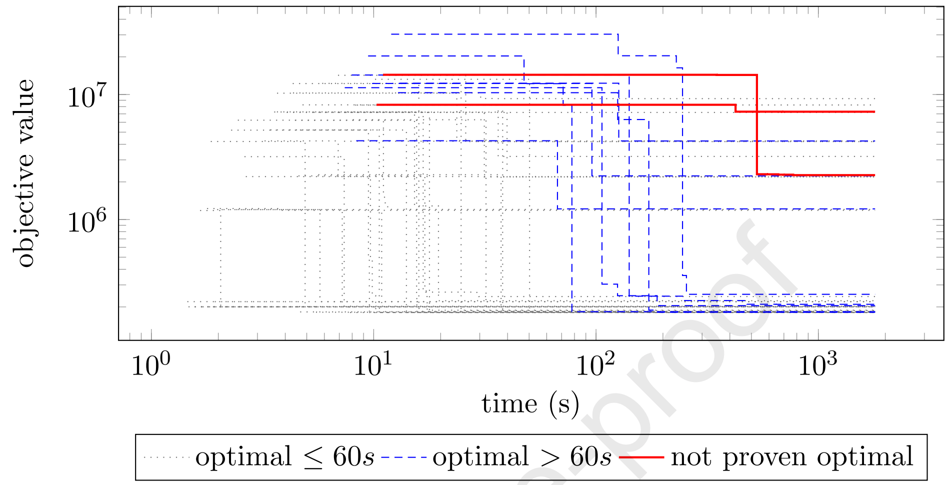


Figure 10: Performance of the exact method over 30 minutes of solving time. Both of the axes are shown on a logarithmic scale. The color (style) of a plot indicates if an instance has been solved to optimality within 60 seconds, after 60 seconds or if no solution could be proven to be optimal within 30 minutes.

7 Conclusion

In this paper, we have introduced a Variable Neighborhood Search heuristic for the rolling stock rescheduling problem. In this heuristic, we use three new neighborhoods. The first neighborhood considers two-opt swaps on the existing rolling stock duties. The second neighborhood improves the assignment of rolling stock for one rolling stock type at a time by making use of the flow properties of this simpler problem. The third neighborhood improves the assignment of composition changes to the transitions. Moreover, we show that a fourth neighborhood can be used to extend the heuristic to rolling stock rescheduling with flexible turning at the terminal stations.

We have tested our heuristic on instances of Netherlands Railways (NS) for both small and large disruptions. Overall, we find that the heuristic is able to provide circulations that are close to the optimal ones for most of the instances. At the same time, we see that for some instances the number of cancellations is larger than found in the optimal circulation, leading to a higher cost for those instances. While we find that an exact method is

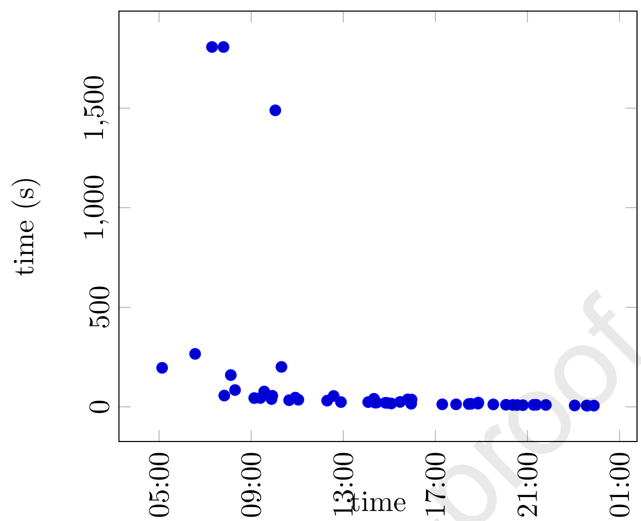


Figure 11: Relation between the start time of the disruption and the running time of the exact solution method.

able to find optimal solutions quickly for rolling stock rescheduling instances without flexible turning, we show that our heuristic is able to outperform an exact method on some of the instances of the largest instance class when including flexible turning. Moreover, we find that the heuristic is generally able to find good solutions quickly, which allows rolling stock dispatchers to terminate the search early if they feel the current solution is of sufficient quality.

Overall, we believe that our results show the potential of local search based techniques for rolling stock rescheduling. Future research may focus on finding additional neighborhoods, as applications in other fields such as vehicle routing have shown that local search heuristics may benefit significantly from considering a wide variety of neighborhoods. Moreover, the heuristic that we have proposed here might benefit significantly from speeding up the way in which negative weight cycles are found in the Adjusted Path Neighborhood.

References

Borndörfer R, Grimm B, Reuther M, Schlechte T, 2017 *Template-based re-optimization of rolling stock rotations. Public Transport* 9:365–383, URL http://dx.doi.org/10.1007/s12469-017-0152-4.

Borndörfer R, Grimm B, Schlechte T, 2019 *Re-optimizing ICE rotations after a tunnel breakdown near Rastatt. RailNorrköping 2019. 8th International Conference on Railway Operations Modelling and Analysis (ICROMA), Norrköping, Sweden, June 17th – 20th, 2019*, 160–168, number 69 in Linköping Electronic Conference Proceedings (Linköping University Electronic Press, Linköpings universitet).

Borndörfer R, Mehrgardt J, Reuther M, Schlechte T, Waas K, 2014 *Re-optimization of rolling stock rotations.* Huisman D, Louwerse I, Wagelmans AP, eds., *Operations Research Proceedings 2013*, 49–55 (Cham: Springer International Publishing), ISBN 978-3-319-07001-8, URL http://dx.doi.org/10.1007/978-3-319-07001-8_8.

Borndörfer R, Reuther M, Schlechte T, Waas K, Weider S, 2016 *Integrated optimization of rolling stock rotations for intercity railways. Transportation Science* 50(3):863–877, URL http://dx.doi.org/10.1287/trsc.2015.0633.

Budai G, Maróti G, Dekker R, Huisman D, Kroon L, 2010 *Rescheduling in passenger railways: the rolling stock rebalancing problem. Journal of Scheduling* 13(3):281–297, URL http://dx.doi.org/10.1007/s10951-009-0133-9.

Cacchiani V, Caprara A, Toth P, 2010 *Solving a real-world train-unit assignment problem. Mathematical Programming* 124(1):207–231, URL http://dx.doi.org/10.1007/s10107-010-0361-y.

Cacchiani V, Caprara A, Toth P, 2013 *A Lagrangian heuristic for a train-unit assignment problem. Discrete Applied Mathematics* 161(12):1707 – 1718, URL http://dx.doi.org/https://doi.org/10.1016/j.dam.2011.10.035, 9th Cologne/Twente Workshop on Graphs and Combinatorial Optimization (CTW 2010).

Cacchiani V, Caprara A, Toth P, 2019 *An effective peak period heuristic for railway rolling stock planning. Transportation Science* 53(3):746–762, URL http://dx.doi.org/10.1287/trsc.2018.0858.

Cacchiani V, Huisman D, Kidd M, Kroon L, Toth P, Veelenturf L, Wagenaar J, 2014 *An overview of recovery models and algorithms for real-time railway rescheduling. Transportation Research Part B: Methodological* 63:15 – 37, URL http://dx.doi.org/10.1016/j.trb.2014.01.009.

Dollevoet T, Huisman D, Kroon LG, Veelenturf LP, Wagenaar JC, 2017 *Application of an iterative framework for real-time railway rescheduling. Computers & Operations Research* 78:203–217, URL http://dx.doi.org/10.1016/j.cor.2016.08.011.

Fioole PJ, Kroon L, Maróti G, Schrijver A, 2006 *A rolling stock circulation model for combining and splitting of passenger trains. European Journal of Oper-*

ational Research 174(2):1281–1297, URL http://dx.doi.org/10.1016/j.ejor.2005.03.032.

Hoogervorst R, Dollevoet T, Maróti G, Huisman D, 2020 *Reducing passenger delays by rolling stock rescheduling. Transportation Science* 54(3):762–784, URL http://dx.doi.org/10.1287/trsc.2019.0968.

Lusby RM, Haahr JT, Larsen J, Pisinger D, 2017 *A branch-and-price algorithm for railway rolling stock rescheduling. Transportation Research Part B: Methodological* 99:228–250, URL http://dx.doi.org/10.1016/j.trb.2017.03.003.

Mladenović N, Hansen P, 1997 *Variable neighborhood search. Computers & Operations Research* 24(11):1097 – 1100, URL http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

Nielsen LK, 2011 *Rolling Stock Rescheduling in Passenger Railways: Applications in short-term planning and in disruption management.* Ph.D. thesis, Erasmus Research Institute of Management, URL http://hdl.handle.net/1765/22444.

Nielsen LK, Kroon L, Maróti G, 2012 *A rolling horizon approach for disruption management of railway rolling stock. European Journal of Operational Research* 220(2):496–509, URL http://dx.doi.org/10.1016/j.ejor.2012.01.037.

Peeters M, Kroon L, 2008 *Circulation of railway rolling stock: a branch-and-price approach. Computers & Operations Research* 35(2):538 – 556, URL http://dx.doi.org/10.1016/j.cor.2006.03.019, part Special Issue: Location Modeling Dedicated to the memory of Charles S. ReVelle.

Reuther M, 2017 *Mathematical optimization of rolling stock rotations.* Doctoral thesis, Technische Universität Berlin, Berlin, URL http://dx.doi.org/10.14279/depositonce-5865.

Veelenturf LP, Kidd MP, Cacchiani V, Kroon LG, Toth P, 2016 *A railway timetable rescheduling approach for handling large-scale disruptions. Transportation Science* 50(3):841–862, URL http://dx.doi.org/10.1287/trsc.2015.0618.

Wagenaar J, Kroon L, Fragkos I, 2017 *Rolling stock rescheduling in passenger railway transportation using dead-heading trips and adjusted passenger demand. Transportation Research Part B: Methodological* 101:140 – 161, URL http://dx.doi.org/10.1016/j.trb.2017.03.013.

Wagenaar JC, Kroon LG, Schmidt M, 2017 *Maintenance appointments in railway rolling stock rescheduling. Transportation Science* 51(4):1138–1160, URL http://dx.doi.org/10.1287/trsc.2016.0701.

- We consider the rescheduling of rolling stock after a disruption occurs

- We propose a Variable Neighborhood Search heuristic for rolling stock rescheduling

- We develop three innovative neighborhoods for local search and perturbation

- We apply the heuristic to real-world instances of Netherlands Railways (NS)

- Solutions of high quality can be obtained within short computation times

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: